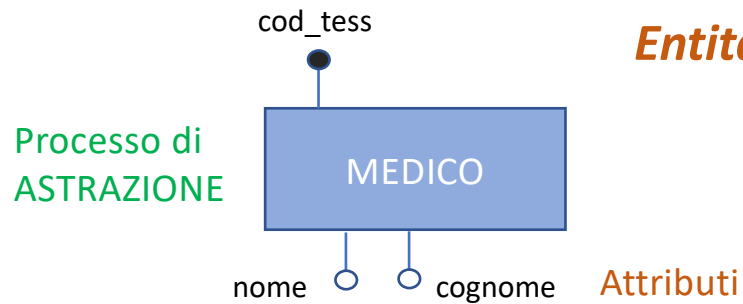


BASI DI DATI Schema/Diagramma ENTITA'-RELAZIONE



Attributi					
codTess	nome	cognome	...		
12345	Alessio	Neri	...		

...		

Una riga
è una istanza,
un dato "concreto"

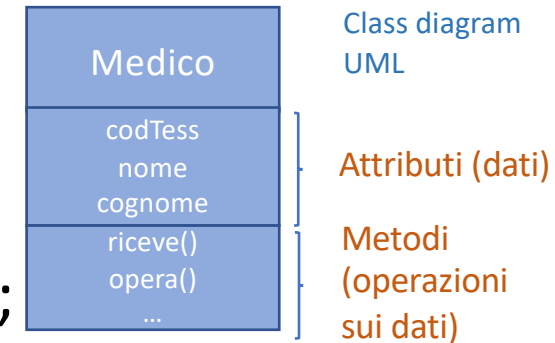
MODELLO RELAZIONALE (TABELLE)

OBJECT ORIENTATION / OBJECT ORIENTED PROGR.

STRUTTURA DEI DATI

```
int x;  
Medico m;
```

Classe

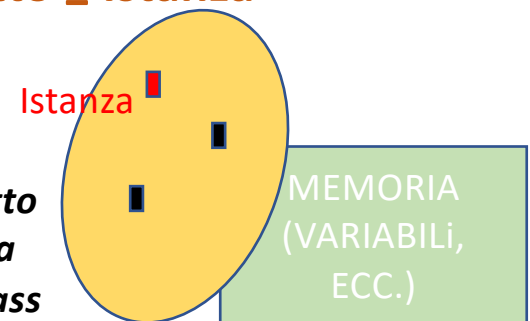


```
x=5;  
m=new Medico();
```

Processo di
REIFICAZIONE/
ISTANZIAZIONE
(inverso
ASTRAZIONE)

Oggetto \equiv Istanza

Un oggetto
è l'istanza
di una class



I DATI SARANNO POI REGISTRATI NELLA
MEMORIA DEL SISTEMA/CALCOLATORE

DATI ("VERI E PROPRI")

PARADIGMA DI PROGRAMMAZIONE PROCEDURALE (PROCEDURE)

E CONCETTI DI BLOCCO, TIPIZZAZIONE E PROGRAMMAZIONE STRUTTURATA

BLOCCO {
int y; // VARIABILE "GLOBALE"

{
int x; // SCOPE o AMBITO DI VISIBILITA'
x = 5; // VAR. VISIBILE SOLO NEL BLOCCO
// IN CUI E' STATA DEFINITA
}

y=5; OK
x=5; ERRORE, VAR. NON VISIBILE FUORI DAL BLOCCO

int z;
z = 4.5; // ERRORE (DI TIPO), PERDITA PARTE FRAZIONARIA

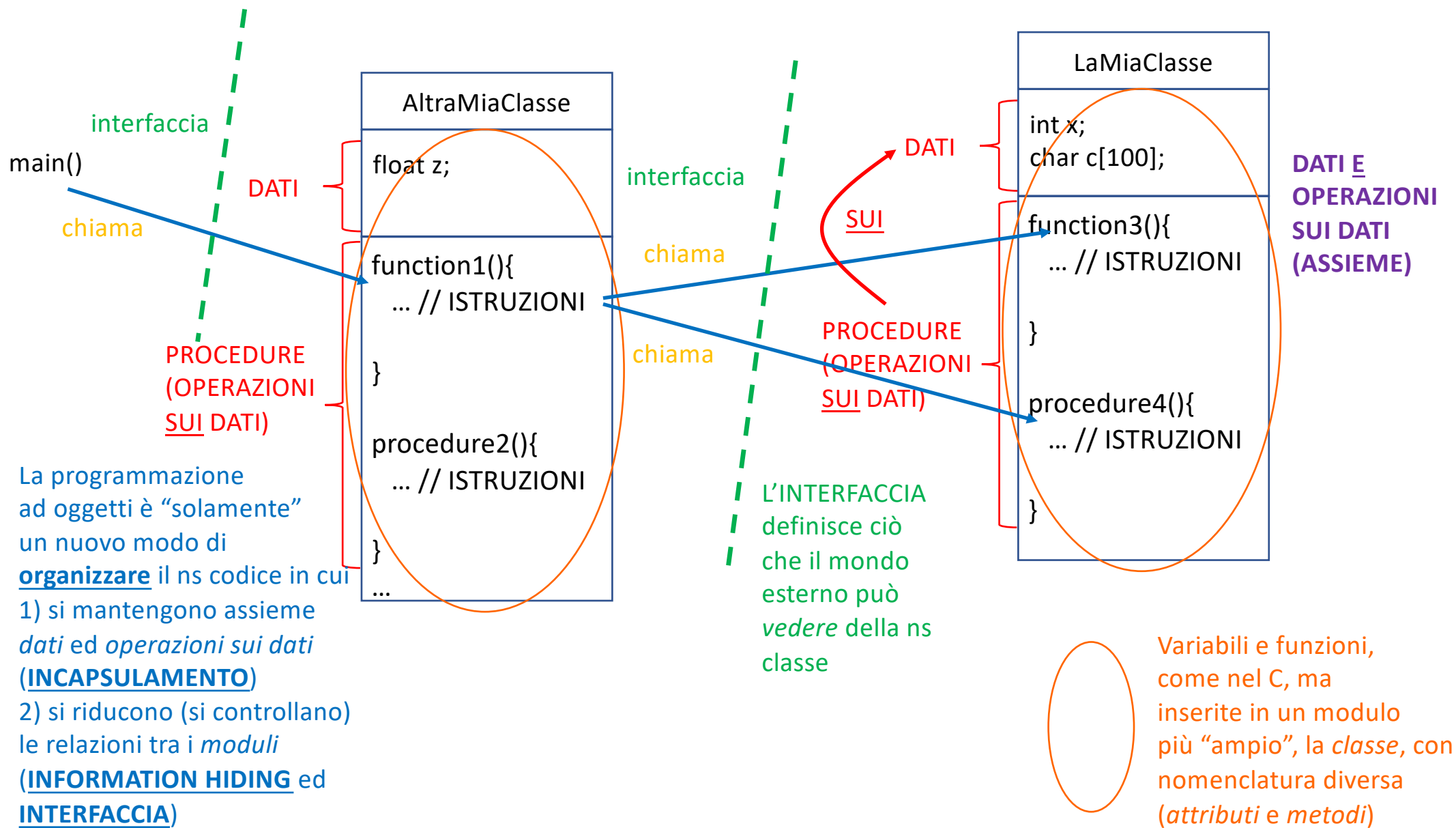
Istruzioni in sequenza
Costrutto di selezione o scelta if() else
Costrutto/i di iterazione for(), while(), ...
Singolo punto di ingresso-uscita dal codice (no salti incond.)

PROCEDURE {
function1(){
... // ISTRUZIONI
}

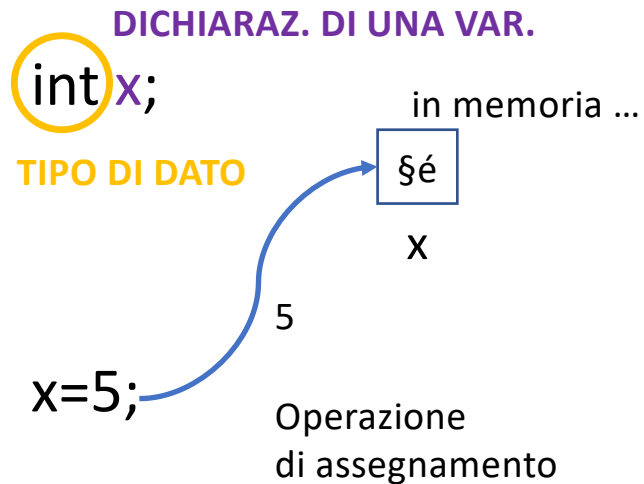
procedure2(){
... // ISTRUZIONI
}

ITERAZ. {
for(...)
{
... // FATTORIZZARE ISTR.
// DA RIPETERE
}

PROGRAM.
STRUTTURATA



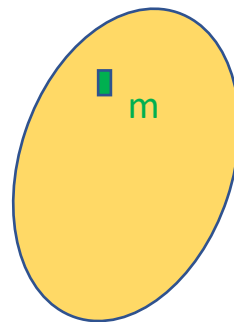
TIPI PRIMITIVI



Medico
codTess
nome
cognome
riceve()
opera()
...

TIPO DI DATO (STRUTTURA DEI DATI)

REIFICAZIONE



DICHIARAZ. DI UNA VAR.

`Medico m;`

m è il NOME dell'OGGETTO
(ma, l'oggetto NON ESISTE ANCORA!)

PERCHE' ESISTA, DEVO CREARLO
(ISTANZIAZIONE)

METODO "COSTRUTTORE"
Costruisce/Crea ed Inizializza
l'oggetto

`m = new Medico();`

HO CREATO UN OGGETTO / UNA
ISTANZA DI CLASSE Medico E
L'HO ASSEGNATO
ALLA VARIABILE m



`m`
VARIABILE

fa riferimento,
"punta"

Medico
12345
Alessio
Neri
riceve()
opera()
...

STATO
Valori
attributi

ISTANZA/OGGETTO
di classe Medico

In memoria avremo gli oggetti creati, anche più di uno, ognuno con un proprio stato ed una propria identità, MA tutti con medesima struttura e funzionalità (operazioni)

DEFINIZIONE/DICHIARAZIONE
DELLA CLASSE Medico

POI CREAZIONE DEGLI
OGGETTI A PARTIRE
DALLA CLASSE

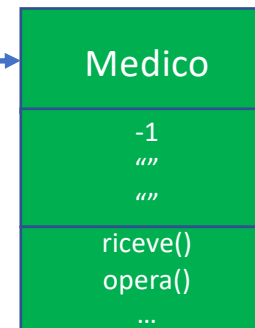
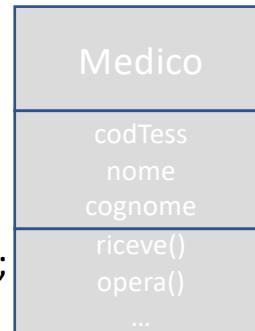
Medico m;

m = new Medico();

INIZIALIZZAZIONE
GARANTITA
DAL COSTRUTTORE
(MA NON E' DETTO CHE
IO SAPPIA QUALE SIA)

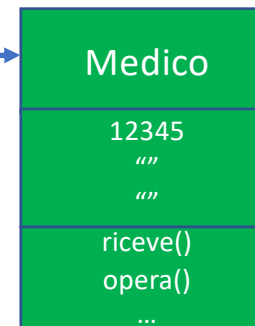
m.codTess = 12345;
m.nome="Alessio";
m.cognome="Neri";
m.opera();

```
class Medico {  
    int codTess;  
    String nome;  
    String cognome;  
  
    void riceve(){ ... };  
    int opera(){ ... };  
}
```



Istanza di (classe) Medico,
creata a partire dalla
struttura definita dalla classe.
Tutte le istanze avranno la
stessa struttura.
Si dice anche
oggetto di classe/di tipo Medico

UTILIZZO DELLA
NOTAZIONE PUNTATA,
CON CUI POSSO
ACCEDERE AD
ATTRIBUTI E METODI
DELL'OGGETTO, ES. PER
INIZIALIZZARE ATTRIBUTI,
INVOCARE METODI, ECC.



Stato dell'oggetto
dopo l'inizializzazione o
dopo che sono stati
assegnati valori ai suoi
attributi

Automobile a1;



a1

Inizialmente non c'è dentro **nulla** di utile
(es., potrebbe essere inizializzata a **null**)

Si può quindi andare a creare un oggetto

Automobile a1;

basic.Automobile@2f0e140b

UID

a1

Unique Identifier dell'oggetto creato

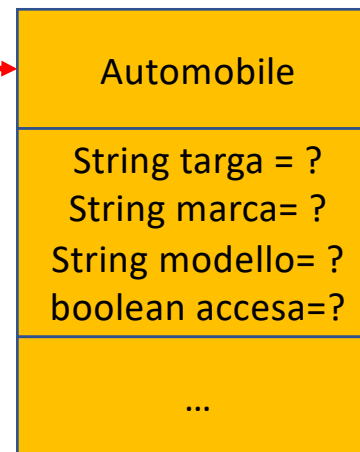
a1 = new Automobile();

System.out.println(a1);

// viene stampato basic.Automobile@2f0e140b

Variabile RIFERIMENTO, con cui si fa riferimento all'area di memoria dell'oggetto
(qualcosa di simile ad un puntatore del linguaggio C)

La "scatolina" appena creata non è abbastanza grande
per ospitare l'oggetto automobile con i suoi dati, ecc.



Automobile a1;

basic.Automobile@2f0e140b

UID

a1

Unique Identifier dell'oggetto creato

a1 = new Automobile();

System.out.println(a1);

// viene stampato basic.Automobile@2f0e140b

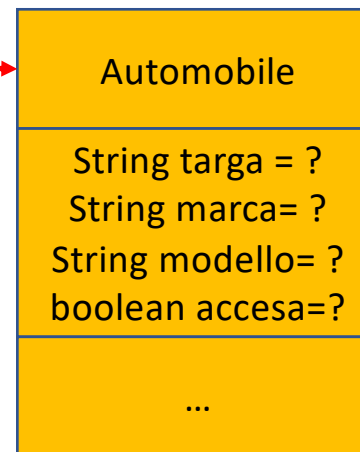
// per stampare targa, marca, ecc.

// occorre accedere ai singoli attributi

System.out.println(a1.targa+" "+a1.marca+" "+...);

Variabile RIFERIMENTO, con cui si fa riferimento all'area di memoria dell'oggetto
(qualcosa di simile ad un puntatore del linguaggio C)

La "scatolina" appena creata non è abbastanza grande
per ospitare l'oggetto automobile con i suoi dati, ecc.



ACCESSO IN LETTURA
AGLI ATTRIBUTI DELL'OGGETTO a1

Variabile RIFERIMENTO, con cui si fa riferimento all'area di memoria dell'oggetto
(qualcosa di simile ad un puntatore del linguaggio C)

Automobile a1;

basic.Automobile@2f0e140b

UID

a1

Unique Identifier dell'oggetto creato

a1 = new Automobile();

System.out.println(a1);

// viene stampato basic.Automobile@2f0e140b

a1.targa="AB123CD";

a1.marca="VW";

...

ACCESSO IN SCRITTURA
AGLI ATTRIBUTI DELL'OGGETTO a1

Automobile

String targa = ?
String marca = ?
String modello = ?
boolean accesa = ?

...

Automobile

String targa = "AB123CD"
String marca = "VW"
String modello = "T Roc"
boolean accesa = "false"

accendi()
vendi()

Variabile RIFERIMENTO, con cui si fa riferimento all'area di memoria dell'oggetto
(qualcosa di simile ad un puntatore del linguaggio C)

Automobile a1;

basic.Automobile@2f0e140b

UID

a1

Unique Identifier dell'oggetto creato

a1 = new Automobile();

System.out.println(a1);

// viene stampato basic.Automobile@2f0e140b

a1.targa="AB123CD";

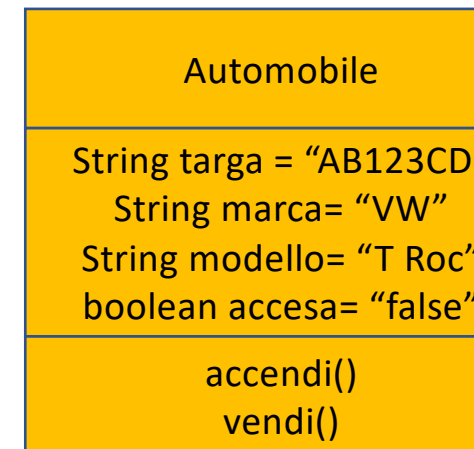
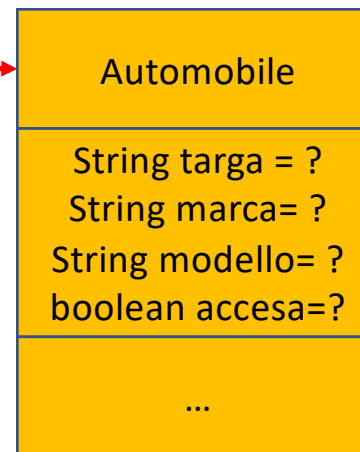
a1.marca="VW";

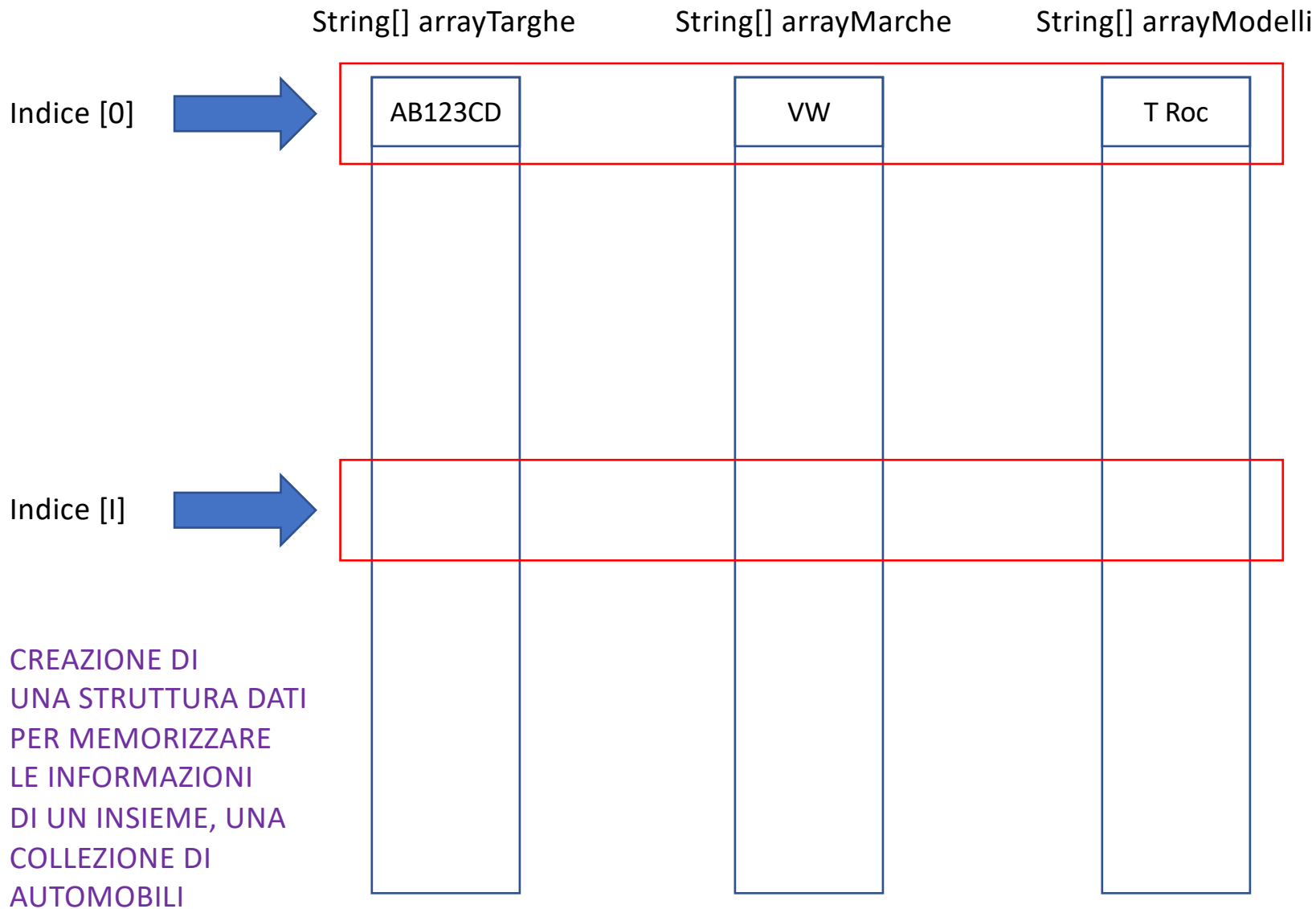
...

a1.accendi();

ACCESSO IN SCRITTURA
AGLI ATTRIBUTI DELL'OGGETTO a1

INVOCAZIONE DEI METODI DELL'OGGETTO ai





Sia usando delle variabili, es. targa1, marca1, ecc., sia usando degli array “paralleli” le informazioni di ciascuna automobile sono distribuite, non si riescono a raggruppare in un unico contenitore

String[] arrayTarghe

String[] arrayMarche

String[] arrayModelli

Indice [0] 

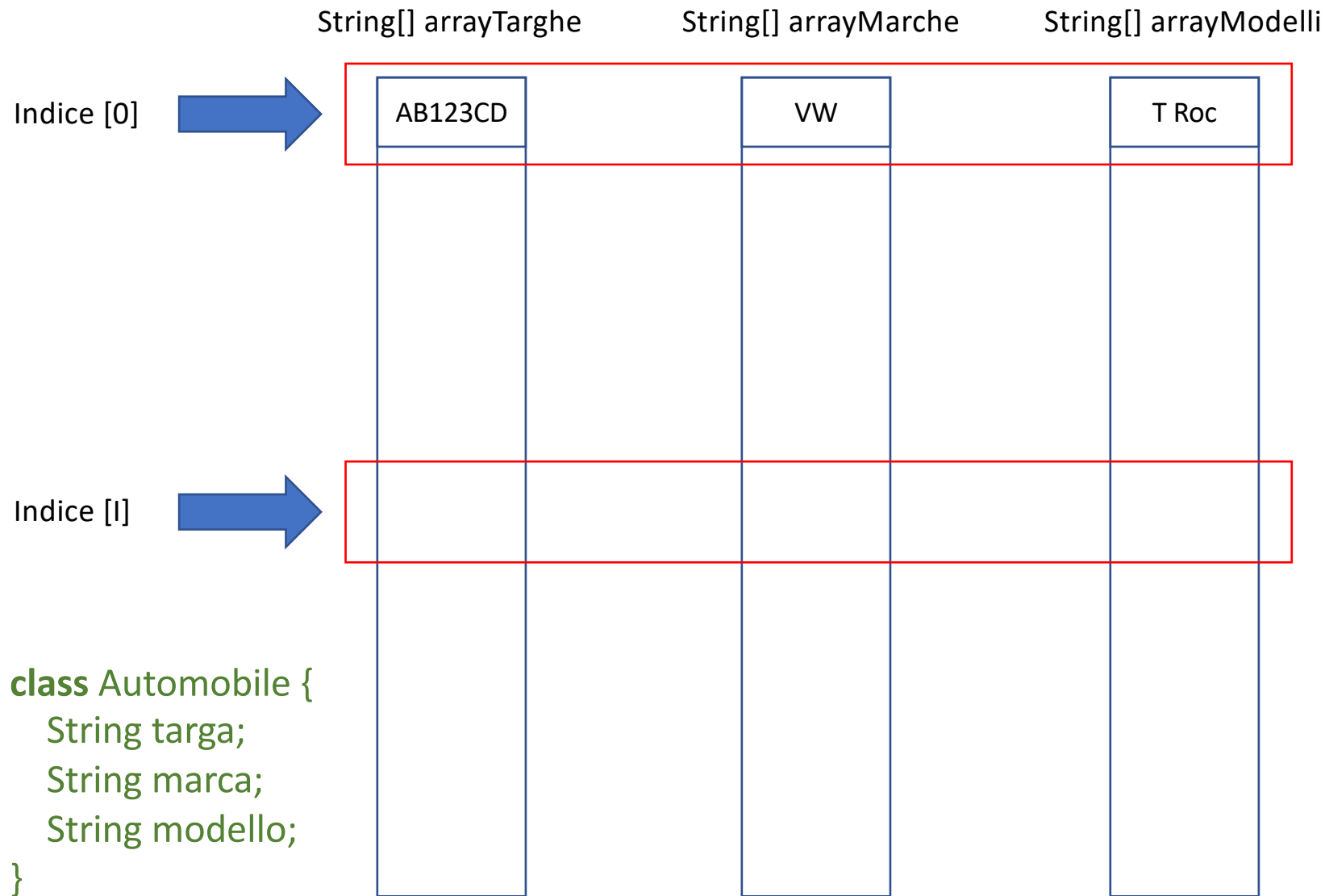
AB123CD	VW	T Roc
---------	----	-------

Indice [I] 

--	--	--

```
struct automobile {  
    char targa[100];  
    char marca[100];  
    char modello[100];  
}
```

In linguaggio C
avrei potuto
pensare di
utilizzare una
struct



In linguaggio C
avrei potuto
pensare di
utilizzare una
struct

In Java, non la
struct (non esiste),
ma la **classe**

Automobile a3;



a1

a3 = new Automobile();

// Stampando a3

Targa: null

Marca: null

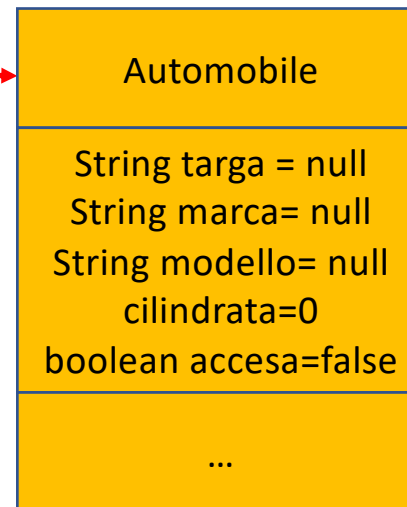
Modello: null

Cilindrata: 0

Accesa: false

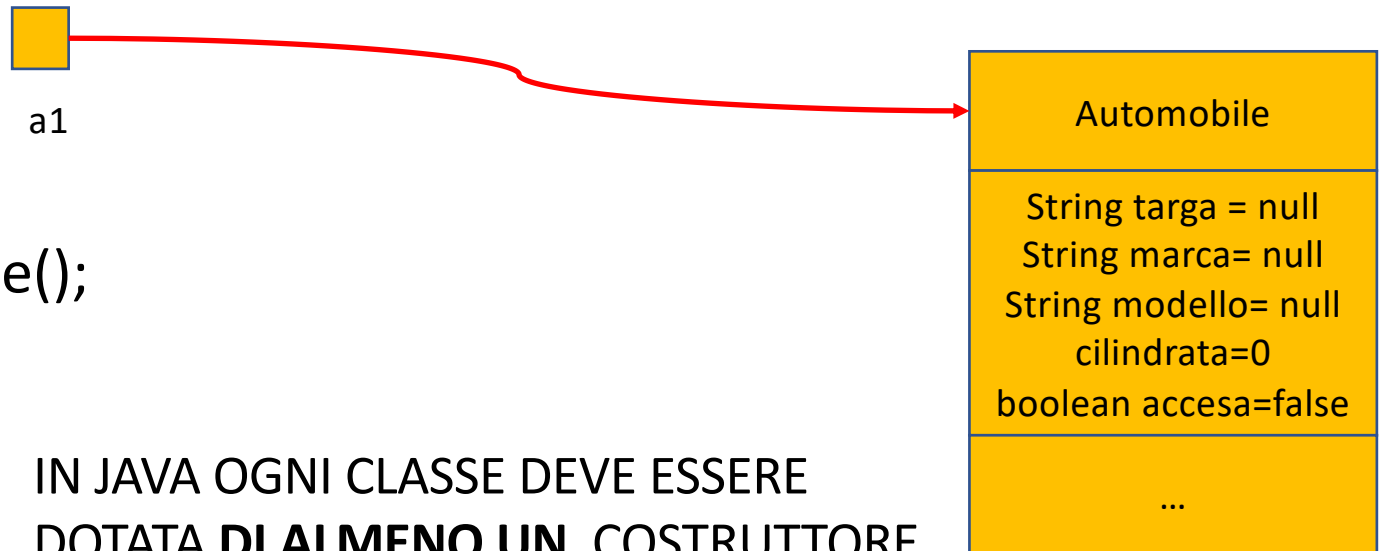
Cosa viene
inserito da Java
nei vari
attributi quando
l'oggetto viene
creato?

**Java li inizializza
con valori
predefiniti**



Numerici a 0 / 0.0
Boolean a false
String a null

Automobile a3;



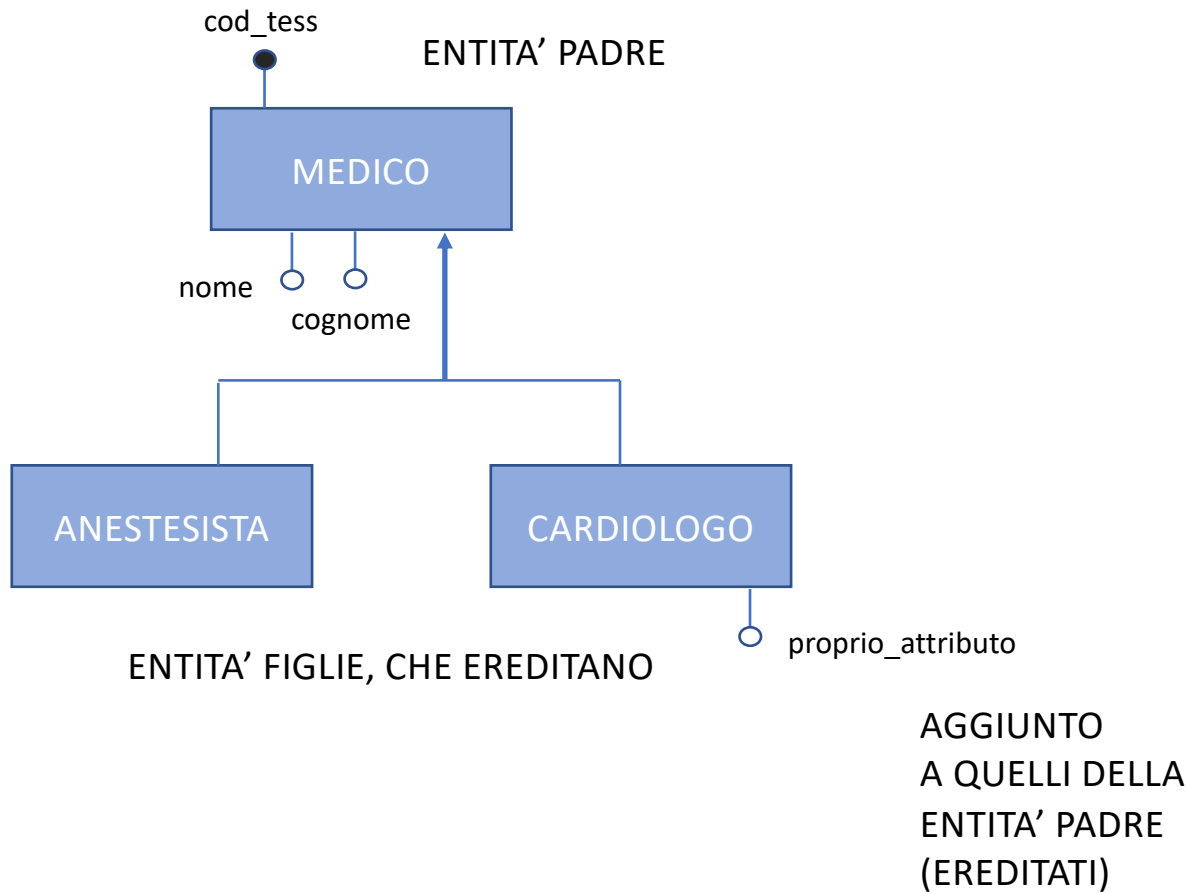
a3 = new Automobile();

**CREAZIONE ED
INIZIALIZZAZIONE**
DELL'OGGETTO
AVVIENE TRAMITE
L'INVOCAZIONE DEL
(METODO)
COSTRUTTORE
Automobile()

IN JAVA OGNI CLASSE DEVE ESSERE
DOTATA **DI ALMENO UN** COSTRUTTORE

COSTRUTTORE **DI DEFAULT** (ANCHE
DETTO "VUOTO" PERCHE' SENZA PARAMETRI

SE IL PROGRAMMATORE NON LO HA SCRITTO,
ESISTE IMPLICITAMENTE, ED **INIZIALIZZA GLI**
ATTRIBUTI AI VALORI PREDEFINITI



EREDITARIETA' NELLE BASI DI DATI

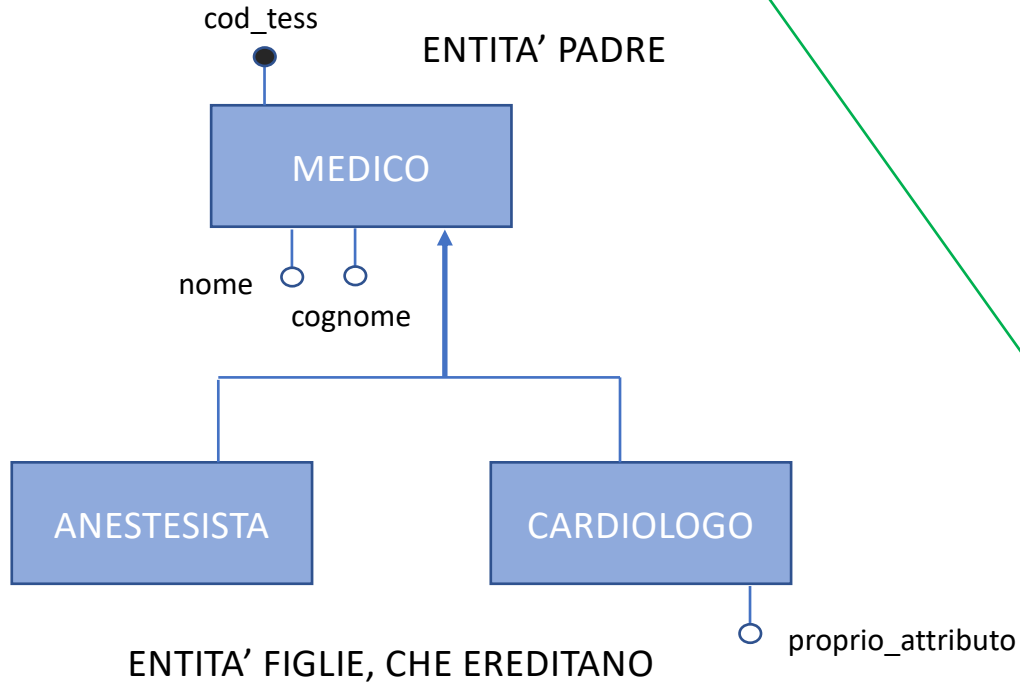
EREDITARIETA' IN OOP

NELL'ORIENTAMENTO AD OGGETTI (INVECE)

Una classe che eredita da un'altra

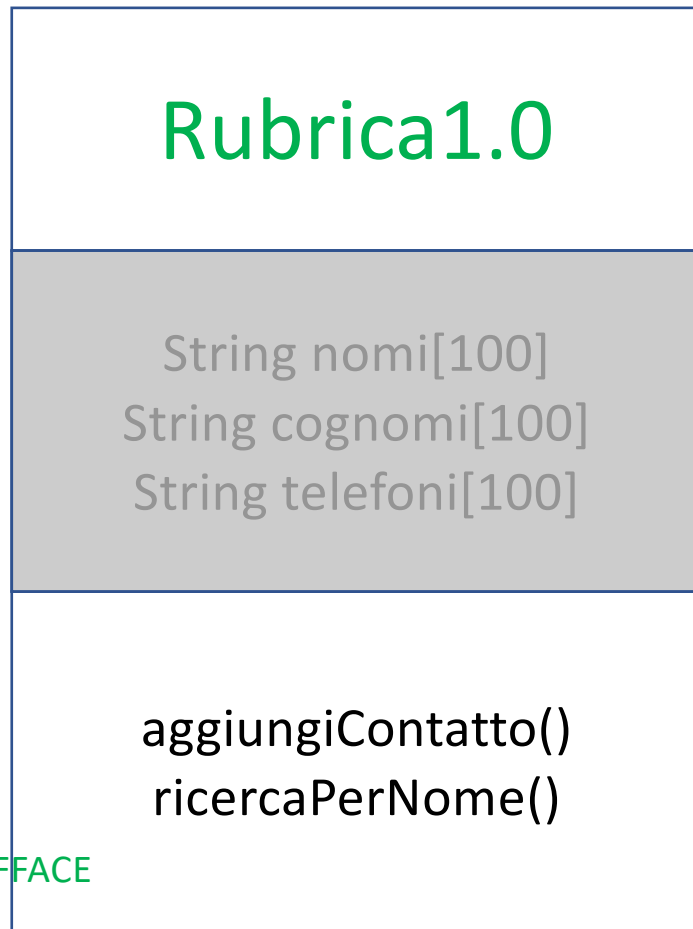
- Ha gli stessi attributi e metodi della classe da cui eredita
- Può **AGGIUNGERE** altri attributi e metodi
- E **PUO' CAMBIARE** quanto ereditato

AGGIUNTO
A QUELLI DELLA
ENTITA' PADRE
(EREDITATI)



EREDITARIETA' NELLE BASI DI DATI

EREDITARIETA' IN OOP



INTERFACE

INTERNALS

