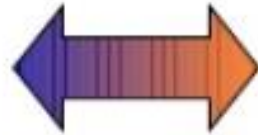


Oggetti

- . **Attributi e Metodi**
- . **Incapsulamento**
- . **Struttura** *degli oggetti*
- . **Interazione** *tra oggetti*
- . **L'interfaccia** *verso l'esterno*

Oggetti



attributi

+

metodi

attributo1
attributo2
attributo3
attributo4

Costituiscono la **memoria** dell'oggetto
e consentono di tenere traccia dello
stato dell'oggetto

metodo1
metodo2
metodo3
metodo4
metodo5
metodo6

Sono le operazioni che un oggetto è in
grado di compiere (**comportamenti**)

Attraverso i metodi un oggetto può
accedere alla sua memoria e
modificare il suo stato



attributi

*Descrivono le proprietà **statiche** dell'oggetto*

*Nella **programmazione** gli attributi vengono realizzati attraverso l'uso delle **variabili** utilizzate dall'oggetto per memorizzare i dati*



metodi

*Descrivono le proprietà **dinamiche** dell'oggetto*

*Nella **programmazione** i metodi vengono realizzati attraverso la scrittura di codice (**procedure** e **funzioni**) che implementano le operazioni dell'oggetto*

L'incapsulamento : afferma che ' l'unico modo di accedere ai dati di un oggetto e' l'invocazione dei suoi metodi, descrive quindi la caratteristica di accesso ai dati per un oggetto

Incapsulamento

La proprietà dell'oggetto di **incorporare** al suo interno **attributi** e **metodi** viene detta **incapsulamento**

L'oggetto è quindi un **contenitore** sia di strutture dati e sia di procedure che li utilizzano

Viene visto come una **scatola nera** (o *blackbox*) permettendo così il **mascheramento dell'informazione** (**information hiding**)

Sezione **Pubblica**

Sezione **Privata**

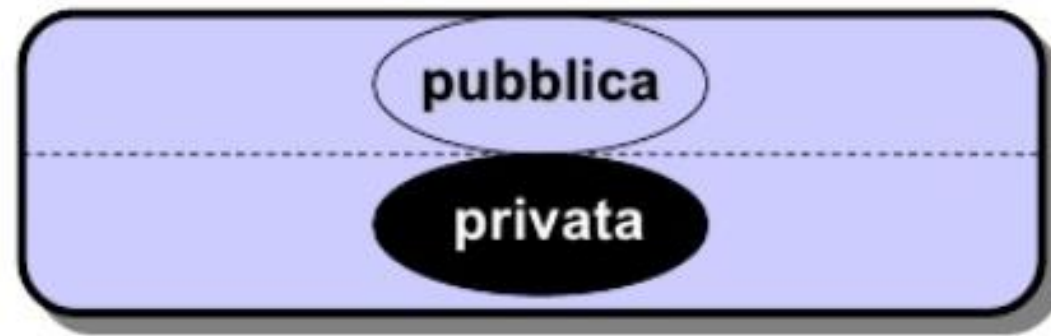
attributi

metodi

attributi

metodi

Struttura degli oggetti



Sezione **Pubblica**

attributi e metodi

che si vogliono rendere **visibili** all'esterno
(e quindi utilizzabili dagli altri oggetti)

Sezione **Privata**

attributi e metodi

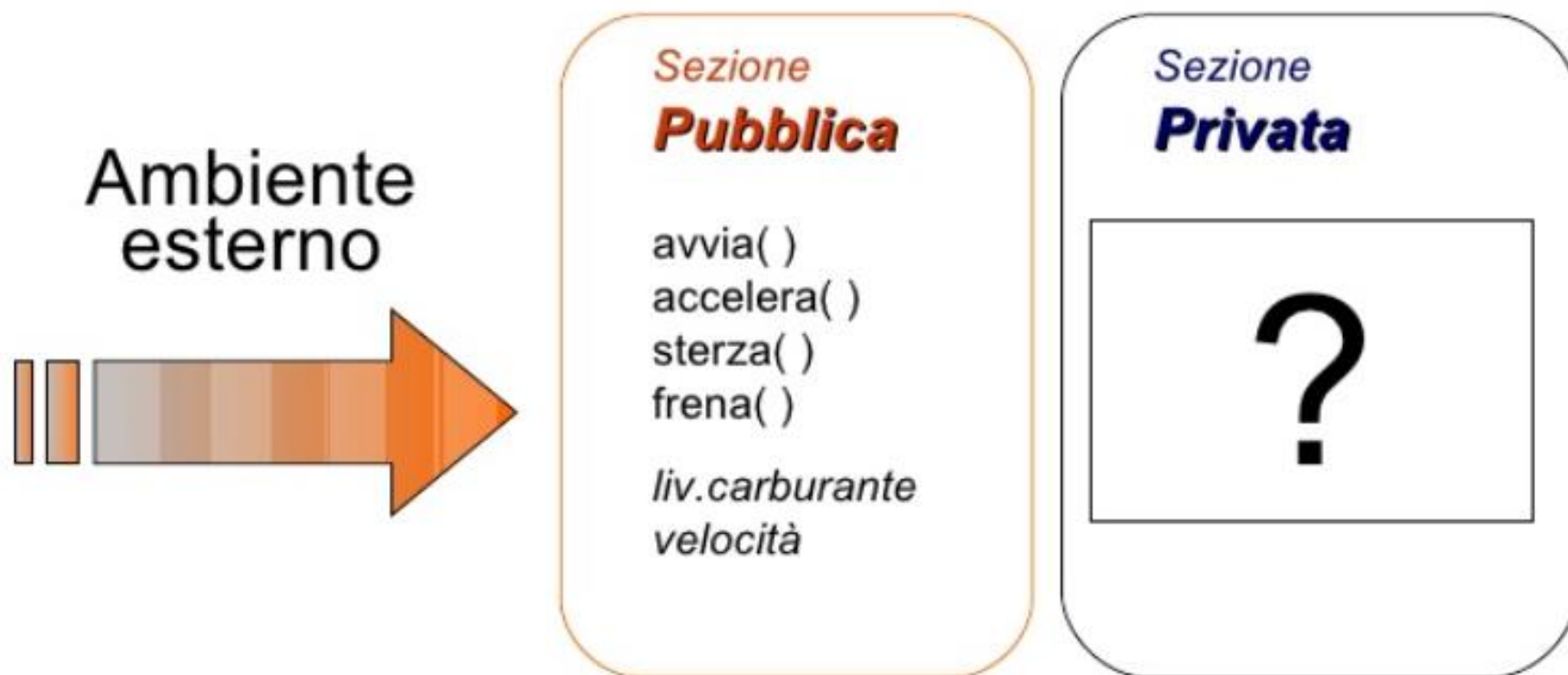
che **non sono accessibili** ad altri oggetti
(e quindi si rendono invisibili all'esterno)

L'interfaccia verso l'esterno

Un oggetto può essere utilizzato ***inviando ad esso dei messaggi***

L'insieme dei messaggi rappresenta ***l'interfaccia*** di quell'oggetto

L'interfaccia ***non consente di vedere*** come sono implementati i metodi, ma ne permette il loro utilizzo e l'accesso agli attributi pubblici



Interazione fra gli oggetti

Un **programma ad oggetti** è caratterizzato dalla presenza di tanti oggetti che **interagiscono** fra loro attraverso il meccanismo dello

scambio di messaggi

Schumacher

Data Nascita = 3/1/69
nazione = **germania**
peso = 74
altezza = 174

Ferrari GA 04

Velocità = 83
colore = **rosso**
liv.carburante = 85,1
posizione marcia = 2

Messaggi:

FerrariGA2004.accelera()

FerrariGA2004.frena()



Metodi:

accelera()

sterza()

frena()

Interazione fra gli oggetti

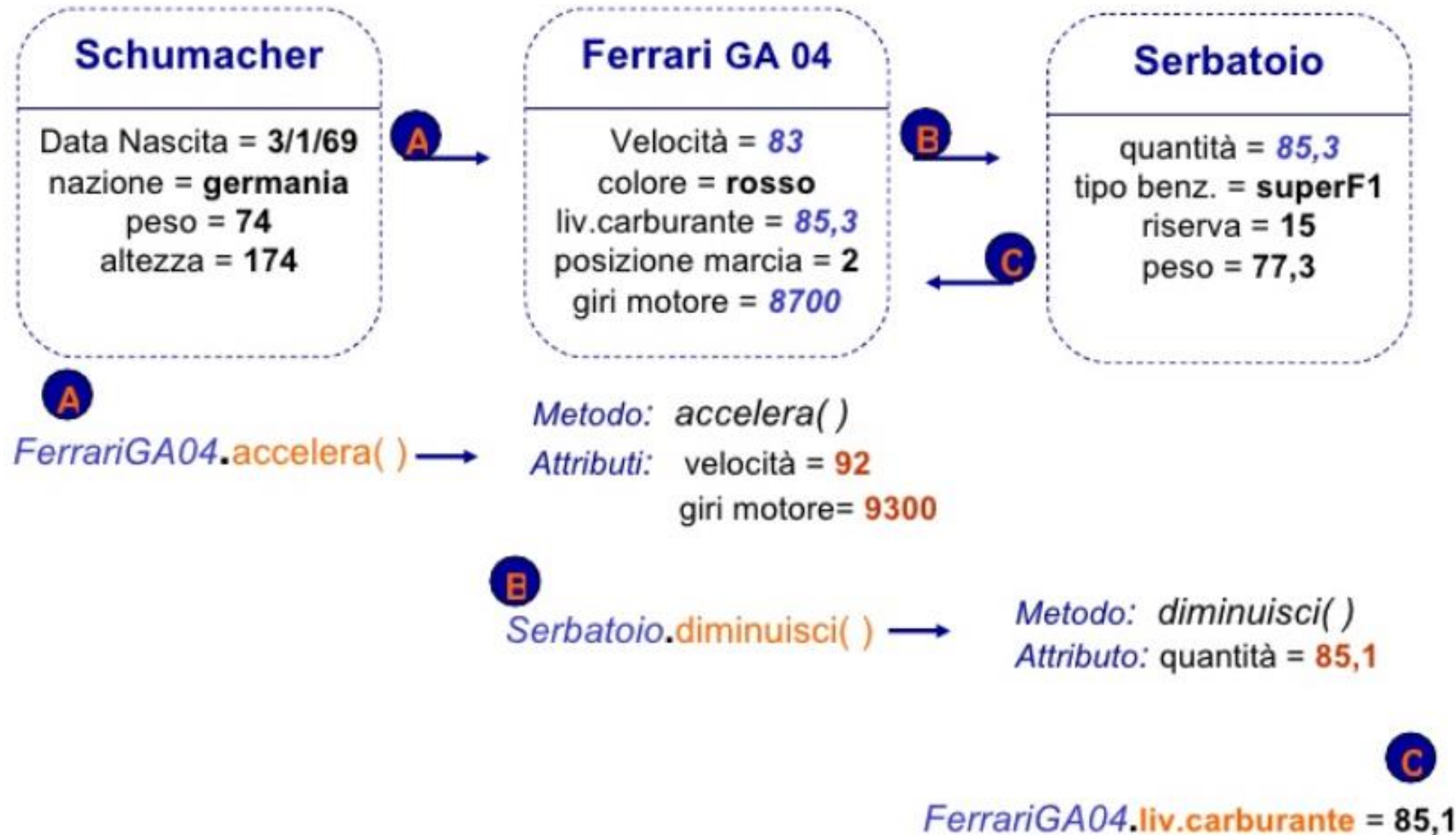
metodi

Quando un oggetto **invoca** un metodo di un altro oggetto ?

- quando vuole **avere un'informazione** sul secondo oggetto
- quando vuole **modificare lo stato** del secondo oggetto

(in sostanza **conoscere** o **modificare** il valore dei suoi **attributi**)

Es. di scambio di messaggi



Nuova Metodologia

Per costruire un programma orientato agli oggetti occorre:

- . Identificare gli **oggetti** che caratterizzano il modello del problema
- . Definire le **classi**, indicando gli **attributi** e i **metodi**
- . Stabilire come gli oggetti **interagiscono** fra loro.

***Linguaggi di programmazione
orientati agli oggetti***

Linguaggi **Puri** —> **ogni cosa è un oggetto**

(...anche un numero intero è definito come oggetto)

- . **Smalltalk**

- . **Eiffel**

Linguaggi **ibridi** —> alcuni tipi di dati **non sono** oggetti

(maggiore libertà a scapito di una maggiore chiarezza)

- . **C++**

- . **Java**

- . **Visual Basic**

- . **Delphi**



```
# include <iostream.h>
class Base {
friend void funzione (Base&);
public:
    int pubblico;
    void Pubblico() {cout <<'Pubblico: ' <<pubblico<<end1;};
private:
    int privato;
    void Privato() {cout <<'Pubblico: ' <<pubblico<<end1;};
protected:
    int protetto;
    void Protetto() {cout <<'Protetto: ' <<protetto<<end1;};
};
void funzione(Base &istanza) {
    istanza.pubblico =1;
    istanza.privato =2;
    istanza.protetto=3;
    istanza.Pubblico();
    istanza.Privato();
    istanza.Protetto();
}
void main() {
    Base istanzabase;
    istanzabase.pubblico =10;
    funzione(istanzabase);
}
```

```
import java.applet.*;
import java.awt.*;
public class semplice extends Applet
{
    private TextArea ta;
    private Label l;
    public void init()
    {
        ta = new TextArea(10,8);
        l = new Label("numeri casuali", Label.CENTER);
        l.setBackground(Color.Yellow);

        setLayout(new BorderLayout());
        add(l, "center");
        add(ta "east");
        generaNumeri();
    }
    public void generaNumeri()
    {
        int casuale;
        for (int i=1; i<=10; i++)
        {
            casuale = (int (Math.random()*1000));
            ta.append("-> "+casuale+"\n");
        }
    }
}
```

```
Private Sub Command1_Click()  
    N = Val(InputBox("Numero di cui vuoi la radice quadrata :", "Radice  
Quadrata"))  
    Msg = "La radice quadrata di " & N  
    SqrN = RadiceQ(N)  
    Select Case SqrN  
        Case 0  
            Msg = Msg & "è zero."  
        Case -1  
            Msg = Msg & "è un numero immaginario."  
        Case Else  
            Msg = Msg & "è" & SqrN  
    End Select  
    MsgBox Msg  
End Sub
```