

## Sommario

Cosa apprenderemo in questo modulo? .....	2
Modulo 1 – Concetti iniziali.....	2
Che cosa è un oggetto?.....	2
Attributi e metodi .....	6
Classe.....	7
Messaggi.....	9
Ricapitolando !.....	10

## Cosa apprenderemo in questo modulo?

- Che cosa è una classe ?
- Che cosa è un attributo ?
- Che cosa è un metodo ?
- Che cosa è un oggetto ?
- Che cosa è un messaggio ?

## Modulo 1 – Concetti iniziali

### Che cosa è un oggetto?

Gli oggetti sono la chiave per comprendere la tecnologia Object Oriented. Se ti guardi intorno, tutto quello che vedi è un oggetto: un'auto, la scrivania, la finestra, un libro, una persona, etc.

Pertanto, il termine oggetto viene utilizzato per rappresentare un elemento del mondo reale. Le caratteristiche essenziali di un oggetto sono:

- **Identità:** ogni oggetto è identificato dalle sue qualità (attributi), dai suoi stati e dai suoi comportamenti.
- **Attributi:** sono le qualità di un oggetto, ad esempio una macchina può avere come attributi: anno, marca, modello, chilometraggio, ecc. Queste qualità possono essere fisse (ovvero non cambiano dalla creazione dell'oggetto) o subire un cambiamento a seconda del comportamento dell'oggetto o del comportamento di altri oggetti esterni.
- **Stato:** rappresenta la situazione in cui si trova l'oggetto in un dato tempo o spazio, ad esempio una macchina può essere ferma o in movimento.
- **Comportamento:** è il servizio che un utente può richiedere all'oggetto, ovvero le azioni che l'oggetto può eseguire, ad esempio, una auto ha come comportamento la capacità di accelerare e frenare.

Gli oggetti possono rappresentare sia i concetti reali (auto, animali, barche, ecc.) e sia concetti astratti (conto corrente, dipendente, persona fisica, ecc.).

## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1

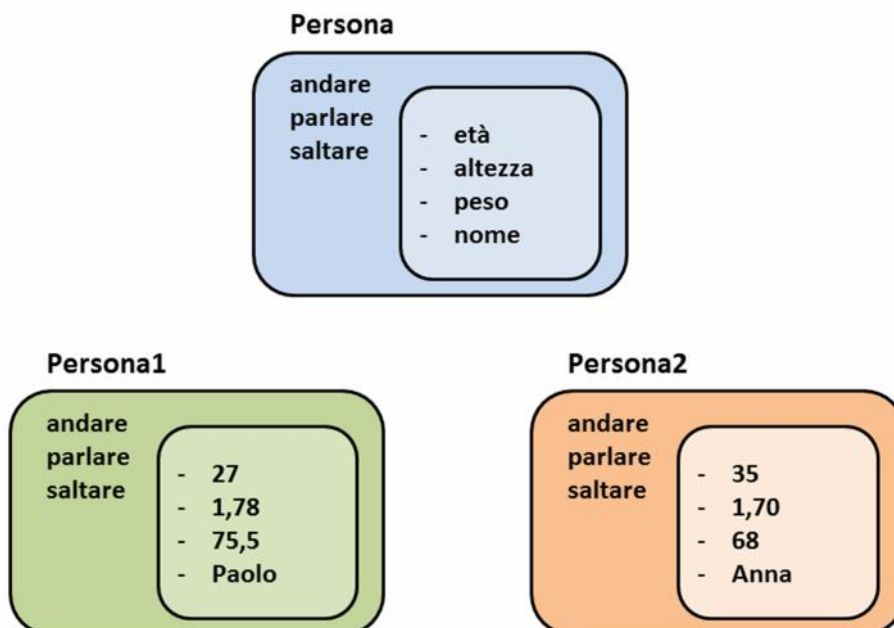


Gli oggetti, di conseguenza, modellano concetti reali o astratti. In un sistema possiamo avere diversi oggetti che interagiscono tra loro. Ogni oggetto può essere replicato in diverse "copie" che rappresentano le **istanze** di tale oggetto. Ad esempio, l'auto 1, l'auto 2 e l'auto 3 sono tutte istanze di oggetti di tipo auto.



L'esempio seguente mostra diverse istanze dell'oggetto Persona. Si noti che ogni istanza di Persona ha un suo diverso particolare stato ed, oltre allo stato, ha anche suoi propri metodi (comportamento) che operano sullo stato stesso. In altre parole, per poter ad esempio saltare, ogni persona impiegherà una certa forza a seconda della sua età, altezza e peso.

L'idea di base è che ogni oggetto è responsabile dei propri dati (stato) ed è in grado di eseguire le operazioni che gli sono state assegnate (comportamento).



## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1

In aggiunta alle caratteristiche essenziali, gli oggetti ed il paradigma Object Oriented portano con sé anche alcuni altri aspetti che possono essere riassunti come segue:

- **Astrazione:** essa rappresenta i diversi punti di vista su di un determinato oggetto, ad esempio un gatto. Nella visione del suo proprietario il gatto è il proprio animale domestico, il suo caro gattino. Dal punto di vista di un veterinario lo stesso gatto è visto come un essere dotato di sistema scheletrico, apparato digestivo, apparato circolatorio, ecc. In altre parole, a seconda del punto di vista, solo gli aspetti salienti ed utili all'analisi del problema vengono considerati importanti.

Di fronte ad un problema complesso, è necessario dividerlo in piccoli pezzi allo scopo di risolverlo più facilmente. Pertanto, in generale, il termine astrazione usato nell'ambito dello sviluppo di sistemi indica che solo gli oggetti che sono effettivamente necessari dovrebbero essere disponibili. Ad esempio, se stiamo guardando la classe persona all'interno di un sistema di pagamento, l'attributo "gruppo sanguigno" non è necessario perché non verrebbe utilizzato in questo ambito. Tuttavia, se il contesto operativo fosse un laboratorio di analisi cliniche, questo attributo dovrebbe probabilmente essere memorizzato.



- **Classificazione:** indica l'attività di raggruppare in una sola classe oggetti che hanno gli stessi attributi, come nell'esempio seguente. I veicoli possono essere raggruppati per colore, per il numero di ruote o per il mezzo su cui si muovono (terra, acqua, ecc).

## PROGRAMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1



- **Incapsulamento:** limita la visibilità delle informazioni di una classe per una maggiore chiarezza, contribuendo a migliorare la manutenzione e favorendo il riutilizzo del codice. Ad esempio, consideriamo una calcolatrice ed un'auto. Possiamo osservare questi oggetti e utilizzarli senza aprirli internamente per vederne il funzionamento.
- **Ereditarietà:** gli oggetti possono essere suddivisi in sottoclassi che "ereditano" gli stessi attributi della superclasse, favorendo il riutilizzo del codice. Ad esempio, consideriamo i mezzi di trasporto. Abbiamo diversi mezzi di trasporto, ma tutti hanno caratteristiche comuni: effettuano trasporto di persone e/o di cose.
- **Polimorfismo:** è inteso come la capacità di un codice di avere diversi comportamenti, oppure la capacità di uno stesso codice di essere utilizzato da più classi. Ad esempio, se analizziamo la classe Poligono, che rappresenta alcune forme geometriche, il metodo di calcolo della sua superficie deve avere comportamenti differenti a seconda del poligono specifico in questione, infatti sappiamo che il calcolo dell'area del Triangolo è diverso dal calcolo dell'area del Quadrato, che sono però entrambi poligoni.
- **Persistenza:** indica che lo stato e il comportamento di un oggetto variano con il tempo. Per esempio, il prezzo di un prodotto.

## Attributi e metodi

Le classi e gli oggetti sarebbero inutili se non fossimo in grado di manipolarli. Attributi e metodi sono il modo orientato agli oggetti per manipolare le istanze di una classe.

Gli attributi sono elementi che definiscono la struttura di una classe. Sono le variabili che memorizzano i tipi di dati specifici come numeri interi, stringhe, numeri reali ma anche dati costituiti da altre classi. Il valore di questi attributi nel tempo indica lo stato di tale oggetto in un determinato momento. Un attributo può essere *di istanza* (ovvero è relativo a quella particolare copia della classe) oppure *di classe*, (cioè relativo alla classe in maniera generale, valido quindi per tutti gli oggetti di quella classe) .

Ecco alcuni esempi di attributi:

```
int età;  
int primo, ultimo;
```

Un **metodo** è come una subroutine, ovvero un blocco di codice che appartiene ad una classe e che serve ad particolare un compito e/o corrisponde ad una azione che l'oggetto può fare. Questo codice è eseguito da un oggetto all'atto della ricezione di un messaggio, e ne determina il suo comportamento. Un metodo è analogo alle funzioni ed alle procedure della programmazione strutturata (simile alle funzioni che troviamo in C ed alle subroutine del Pascal).

Poiché i metodi rappresentano delle azioni, sono generalmente identificati da verbi. Nell'esempio di una classe che rappresenta un animale, i possibili metodi associabili alla classe sono "mangia()", "corri()", "dormi()", ecc. L'utilità dei metodi è di evitare inutili copie di codice, essendo sufficiente richiamare i metodi quando lo si desidera. Inoltre, è possibile riutilizzare il codice di classi esistenti e velocizzare in tal modo l'implementazione del sistema finale.

Di seguito è riportato un esempio di una classe denominata Calcolatrice che implementa operazioni matematiche di base tra due numeri passati come parametri ai metodi:

```
public classe Calcolatrice {  
    public int sommare(int num1, int num2 ){  
        restituire num1 + num2;  
    }  
  
    public int sottrarre(int num1, int num2 ){  
        restituire num1 - num2;  
    }  
  
    public int moltiplicare( int num1, int num2 ){  
        restituire num1 * num2;  
    }  
  
    public int dividere( int num1, int num2 ){  
        restituire num1 / num2;  
    }  
}
```

E' importante notare che la programmazione orientata agli oggetti prevede che attributi e metodi possano avere regole per l'accesso o la visibilità. Ma come facciamo a controllare l'accesso agli attributi e metodi in

## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1

una classe? Questa operazione è semplice: si utilizzano parole riservate come **private**, **public** e **protected**. Nei linguaggi di programmazione troviamo le seguenti keyword "*private*" (privato) , "*public*" (pubblico) e "*protected*" (protetto).

Modificatore di accesso	Descrizione
public	Gli attributi e metodi pubblici sono sempre accessibili a tutti i metodi di tutte le classi. Questo è il livello meno rigido di incapsulamento che equivale a non incapsulare.
private	Gli attributi e metodi privati sono disponibili solo ai metodi (tutti) della classe stessa. Questo è il livello di incapsulamento più rigido.
protected	Gli attributi e metodi di questo tipo sono disponibili ai metodi della classe stessa ed alle sue sottoclassi (si veda il capitolo sull' Ereditarietà).
Non specificato, equivale a "default"	Gli attributi ed i metodi sono disponibili in questo caso solo ai metodi di classi appartenenti allo stesso "pacchetto" ( <i>package</i> ). Questa modalità di accesso è anche detta " <i>friendly</i> " in alcuni linguaggi di programmazione.

Nota: I modificatori "private" e "public" possono essere anche utilizzati con le classi.

### Classe

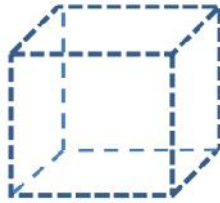
Classe è un termine usato nei linguaggi Object Oriented per descrivere un insieme di elementi con determinate caratteristiche comuni. Una classe può essere intesa come uno "scheletro" che descrive le caratteristiche (**attributi**) e comportamenti (**metodi**) di elementi che compongono un sistema. Questo punto è fondamentale per comprendere la seguente distinzione:

- la **Classe** è la descrizione generale (implementazione) di un oggetto, con attributi e metodi;
- l'**Oggetto** è una istanza di una classe, con valori specifici per i suoi attributi;

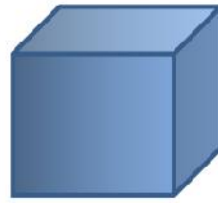
Pertanto una classe è un insieme di oggetti, appartenenti ad un sistema, che condividono le stesse caratteristiche e le stesse operazioni. Vedere l'esempio di seguito per meglio comprendere la differenza tra classe ed oggetto:

## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1

Esempio di una classe chiamata "Cubo".  
La classe definisce le caratteristiche ed il comportamento che l'oggetto dovrebbe avere.



Esempio di un oggetto della classe "Cubo".  
Un oggetto è una istanza di una classe. Questo significa che l'oggetto creato segue le caratteristiche ed i comportamenti della classe di origine.



Un'altra caratteristica che differenzia le classi e gli oggetti è data dal fatto che le classi sono concetti facilmente identificabili nella specifica dei sistemi, spesso descritti come sostantivi. Gli oggetti sono legati al mondo reale, e rappresentano istanze di classi (copie di classi) e che si trovano nella memoria di un programma durante l'esecuzione. Invece **la classe è un concetto statico**, cioè, una volta impostato rimane come è nel programma mentre **l'oggetto è un concetto dinamico**, il che significa che può essere creato, modificato e cancellato nella memoria, mentre il programma è in esecuzione.

Per **definire una classe** abbiamo bisogno di :

- Il nome della classe;
- Il nome della sua classe madre (i dettagli sul modulo dell'ereditarietà);
- Il nome e il tipo delle variabili (attributi);
- Il nome e le specifiche delle sue funzioni (metodi).

Vediamo un esempio:





## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1

La classe di cui sopra può essere istanziata, per esempio, con i seguenti oggetti:

```
Cubo cuboA;
```

```
Cubo cuboB;
```

In termini di implementazione, la definizione della classe e la creazione delle istanze di oggetti può essere fatta nel seguente modo:

```
public classe Cubo {  
    String colore;  
    double bordo;  
  
    getBordo(){ ... };  
    calcolaVolume(){ ... };  
}
```

Quando definiamo una classe in un sistema, di solito specifichiamo il comportamento che classe dovrebbe avere attraverso la dichiarazione dei suoi metodi (con nome, la lista dei parametri in ingresso ed il tipo restituito). Inoltre, si è soliti descrivere anche come lo stato della classe verrà modificato attraverso i metodi. In alcune situazioni, non ci interessa "catturare" il comportamento di una classe già al momento della sua creazione, ma dobbiamo avere la possibilità di definire poi successivamente quale sarà in dettaglio il comportamento della classe.

Ad esempio, supponiamo di voler modellare una macchina per il caffè (Caffettiera): non importa come la macchina del caffè funzionerà in dettaglio, ma dobbiamo fare in modo che coloro che la implementano inseriscano un metodo "faiCafe ()" che prende in input i chicchi di caffè e l'acqua, e che produce in output il caffè. In questo caso, nel definire la classe Caffettiera, abbiamo appena reso esplicito il metodo "faiCafe()" e definito anche la sua firma, ma non lo abbiamo implementato, lasciando il dettaglio da impostare in futuro quando si avranno, ad esempio, informazioni su marche specifiche di caffè (altri dettagli nel modulo sull'ereditarietà).

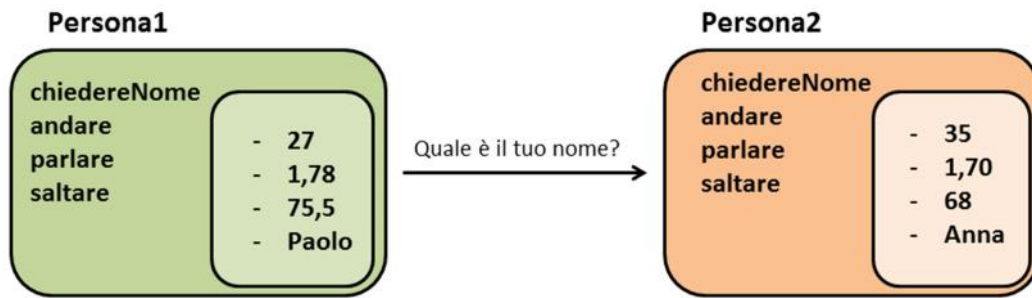
In questo modo, diciamo che "faiCafe()" è un metodo **astratto**, ossia definisce l'interfaccia con il mondo esterno ma lascia l'implementazione del suo comportamento ad una fase successiva. Quando una classe ha almeno un metodo astratto, è detta astratta e non può essere istanziata (ossia non è possibile generare oggetti di quella classe in memoria), fino a che tutti i suoi metodi astratti abbiano una effettiva implementazione.

### Messaggi

Un oggetto isolato all'interno di un sistema non ha alcun significato. Affinché un oggetto abbia senso deve essere possibile metterlo in relazione con altre entità del sistema, consentendone lo scambio di informazioni e l'elaborazione dei dati.

Come affermato in precedenza, la comunicazione tra gli oggetti si ottiene attraverso i messaggi, che corrispondono a chiamate di metodi pubblici di classi. Così, quando un oggetto "A" vuole comunicare con un oggetto "B", "A" invia un messaggio a "B", oppure, "A" esegue un metodo pubblico implementato in "B".

## PROGRAMMAZIONE ORIENTATA AGLI OGGETTI - MODULO 1



Pertanto, un messaggio per essere efficace, deve essere composto da tre parti:

- Un oggetto di destinazione che riceverà il messaggio;
- Il nome del metodo che deve essere invocato dal messaggio;
- I parametri in ingresso al metodo richiamato;

Si noti che in aggiunta a questi tre aspetti del messaggio è possibile anche aggiungere la risposta che il metodo invocato può eventualmente restituire.

### Ricapitolando !

In questa sezione abbiamo imparato dei concetti fondamentali della programmazione orientata agli Oggetti. Prima di passare ai prossimi moduli, fare in modo che i concetti di classe, attributo, metodo, oggetto e di messaggio siano chiari.