


## Corso base - Lezione 3

[http://www.vbsimple.net/base/bas\\_03.htm](http://www.vbsimple.net/base/bas_03.htm)


- [Introduzione alla programmazione ad oggetti.](#)
- [Le classi e le istanze.](#)
- [Gli eventi.](#)
- [I moduli.](#)

Il concetto di [programmazione ad oggetti](#) risale ai primi anni '80. È una tecnica alternativa alla normale programmazione modulare.


La programmazione ad oggetti si fonda sul concetto di [classe](#) , un modello di dati che può essere sfruttato per generare tanti esempi di strutture dati. Per esempio esiste una classe animale a cui appartengono tutti gli animali del pianeta. La classe animale serve per identificare e *costruire* tutti gli animali.

La classe è la base della programmazione ad oggetti. Una classe è una struttura che al suo interno contiene degli elementi detti [membri](#). Questi possono essere semplici dati o funzioni interne.

Trasformiamo tutta questa teoria in esempi pratici? Ok.

Una classe è un esempio di dati che deve essere usato per generare degli elementi di questa natura. Una classe **motocicletta** è un modello per tutte le motociclette. Dentro la classe motocicletta abbiamo dei parametri detti [proprietà](#) . Ogni motocicletta ha un suo colore, una velocità massima, un tipo di sedile, ecc...

Ma ogni tipo di motocicletta è diverso da un'altra. La classe motocicletta è la base su cui si basano tutte le motociclette. I singoli esemplari di motocicletta sono delle copie del modello base. In programmazione il modello si chiama classe, mentre i singoli esemplari si chiamano [oggetti](#) e sono [istanze](#) della classe. Le caratteristiche di un oggetto (di una singola moto) non influenzano minimamente le caratteristiche degli altri oggetti. Il colore di una moto rossa non ha nulla a che vedere con il colore di una moto verde. La caratteristica del colore è in programmazione una proprietà della classe. Ogni oggetto avrà le sue proprietà, indipendenti dagli altri oggetti.

Ma ogni motocicletta ha anche delle funzioni comuni a tutte le motociclette: l'atto dell'accelerare, del frenare, del fermarsi, ecc... Il corrispondente di questi atti in programmazione si chiama funzioni, e specificatamente nella programmazione ad oggetti si chiama [metodo](#) . Come le proprietà anche i metodi sono indipendenti dagli altri esemplari di oggetto. Sarebbe assurdo se, quando una motocicletta frena, tutte le motociclette del mondo frenassero.

Sia le proprietà che i metodi di una classe sono i membri contenuti in tale classe. Per accedere da Visual Basic ad una proprietà o ad un metodo è necessario fornire il nome

dell'oggetto a cui ci si riferisce seguito da un punto e dal nome del membro (proprietà o metodo). Tuttavia alcune proprietà o metodi non possono essere richiamati direttamente, ma bisogna seguire delle scelte implementative obbligatorie. Per esempio non esiste un modo di passare dalla velocità 50 Km/h a 150 Km/h senza usare l'acceleratore. Non esiste un modo di modificare la velocità direttamente, ma ci sarà un metodo che si occuperà di fare le operazioni giuste per accelerare. La velocità può essere considerata una proprietà di **sola lettura**, poichè sarà possibile leggerne il valore sul cruscotto, ma non sarà possibile modificarla direttamente. In programmazione esistono anche proprietà di **sola scrittura** e proprietà leggibili e scrivibili.

Visual Basic, così come altri linguaggi, usa collegare delle decisioni all'intervenire di certe situazioni. Per esempio alcuni modelli di moto al momento dell'accensione delle luci direzionali fanno un suono per avvertire di questo stato. Ma questo non avviene non per tutte le moto. Come fare a gestire queste situazioni?

In programmazione allo scatenarsi di certe situazioni (come l'accensione delle luci direzionali) viene eseguito un [evento](#) ⚡. Poi i singoli oggetti reagiscono in maniera differente ad ogni evento. Il programmatore può e deve sfruttare gli eventi provenienti dall'oggetto per operare di conseguenza. Nell'esempio precedente all'atto dell'accensione delle luci viene lanciato un evento di nome LuciAccese. Il produttore della moto può decidere se una moto deve fare qualcosa in particolare nella situazioni di luci accese. Nel nostro caso faremo produrre al cruscotto un suono di avvertimento.

Le classi possono essere anche essere usate come modello per crearne altre, più specifiche o in parte diverse, al fine di sviluppare oggetti di genere diverso. Questa operazione di replica di classi è detta [subclassing](#).

Un concetto abbastanza diverso dalla classe è il [modulo](#) 🧩. Quest'ultimo può contenere valori e funzioni libere, che possono essere sfruttate da chiunque e non soltanto da un oggetto. Un esempio pratico si può fare con i semafori, i quali funzionano sempre e comunque, in presenza di traffico, sia in assenza di movimento. Il semaforo funziona sempre ed in ogni caso e chiunque può guardare il suo stato, sia i motociclisti, sia gli automobilisti, sia i passanti.

I dati presenti in un modulo sono tipicamente globali, cioè accessibili a qualunque parte del programma, sebbene sia possibile restringere questo campo a determinate parti del programma.

Affrontati questi argomenti, propedeutici per una buona comprensione dei meccanismi di Visual Basic, possiamo affrontare serenamente e con divertimento la pratica delle nozioni finora imparate.

[Fibia FBI](#)  
24 Ottobre 2000



[Torna alla seconda lezione](#)

[Vai alla quarta lezione](#)

