PROGRAMMAZIONE OBJECT ORIENTED: LE CLASSI

Nuove Classi in Visual C#

(esempio pilota: ClasseFrazioni)

In un linguaggio di programmazione, una **Classe** è un *modello* usato per creare uno o più **Oggetti** dello stesso tipo. Gli Oggetti creati sulla base di una stessa Classe si chiamano **Istanze** della Classe.

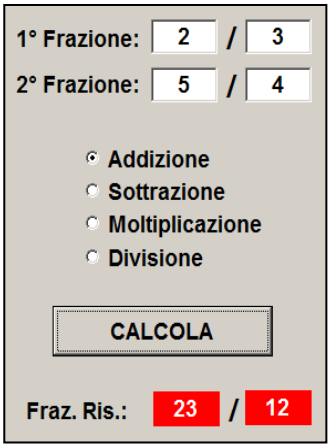
☞ Grazie alla *Classe TextBox* è possibile disporre su una form, tanti oggetti di tipo *Casella di Testo* e chiamarli txtA, txtB, txtRis, eccetera. Gli oggetti *txtA, txtB, txtRis* sono *Istanze* della *Classe TextBox*.

C#, come tutti i Linguaggi di Programmazione **Orientati agli Oggetti** *(OOP)*, permette al programmatore di definire **Nuove Classi** e di creare di conseguenza dei propri Oggetti "*personalizzati*".

☞ Nel nostro *esempio* definiremo una *Nuova Classe* chiamata *Frazione*: essa ci consentirà di creare *Istanze* (cioè Oggetti), ciascuna delle quali sarà adatta a *memorizzare e gestire una Frazione*.

Una Classe viene creata con una **Pagina di Codice** che definisce la **Struttura Dati** interna, i **Costruttori**, le **Proprietà**, i **Metodi** e gli **Eventi** che caratterizzeranno gli Oggetti dalla Classe.

☞ Una *Istanza FA* (cioè l'*oggetto FA*) della nostra nuova classe *Frazione*, per gestire una frazione, dovrà offrire: dei *Costruttori* per creare la frazione, delle *Proprietà* (come *Numeratore* e *Denominatore*) per permettere di impostare la frazione; dei *Metodi* (come *Somma*, *Prodotto*, *ecc.*) per consentire di effettuare operazioni sulla frazione; ecc.



Creazione di Oggetti e Metodo Costruttore

☞ L'esempio prevede di realizzare un programma che effettui delle *Operazioni sulle Frazioni*.

☞ Vengono chieste in input *2 Frazioni* (la frazione **FA,** con le caselle **txtNumA** e **txtDenA** e la frazione **FB,** con le caselle **txtNumB** e **txtDenB**) e il *tipo di operazione* da effettuare (con degli appositi *RadioButton).* Infine, con un *Pulsante*, si effettua il calcolo, ponendo il risultato in una *terza frazione* (**FR**) e visualizzandola nelle label **lblNumR** e **lblDenR.**

✋ Inizialmente supponi di aver *già sviluppato* la **classe Frazione** e che essa sia *già funzionante*: impariamo anzitutto ad *utilizzarla* e a scrivere codice per *creare istanze* di classe Frazione e *richiamarne i metodi*.

Prima di "*creare*" un oggetto (istanza) della Nuova Classe, è necessario **Dichiarare l'Istanza** stessa. La dichiarazione avviene *come se si dichiarasse una normale variabile*, indicando, come "tipo", la classe stessa:

<classe> <nome-istanza>	*… dichiarazione dell'istanza …*

☞ Per dichiarare un'istanza (cioè un oggetto) di nome FA e di classe Frazione, scrivi:

Frazione FA

… proprio come se volessi dichiarare una "variabile" di "nome" FA di "tipo" Frazione.

Dopo averla dichiarata, per **Creare una Istanza** di una Nuova Classe, si utilizza la **Clausola new** seguita dal cosiddetto **Metodo Costruttore** della Classe, secondo la regola:

<nome-istanza> = **new** *<costruttore>* **(** *<parametri>* **)**	*… creazione dell'istanza …*

Il *Costruttore* ha sempre *lo stesso nome della classe* e i **parametri**, separati da virgola, indicano al costruttore *i dati necessari per inizializzare* e, quindi, "*costruire*" l'oggetto.

☞ Ad esempio, per creare un oggetto Frazione contenente la **frazione 4/5**, scrivi: **FA = new Frazione (4, 5);**

… "Frazione (4, 5)" è il *Metodo Costruttore* (stesso nome della classe) e, fra parentesi, sono indicati il *Numeratore* e il *Denominatore*, ossia *i dati necessari a inizializzare l'oggetto*, in modo che si tratti della frazione 4 / 5.

E' possibile **Dichiarare e Creare** l'istanza, *in un'unica istruzione*:

<*classe*> <*nome-istanza*> = **new** <*costruttore*> **(** <*parametri*> **)**	*… dichiarazione + creazione …*

☞ Puoi dichiarare e creare in un sol colpo, scrivendo: **Frazione FA = new Frazione (4, 5);**

☞ Procediamo ora alla scrittura del codice del nostro programma di esempio. Nella form, devi programmare l'evento *click* di *plsCalcola* e "creare" una nuova Frazione FA utilizzando i dati inseriti nelle caselle di testo txtNumA e txtNumB:

```
private void plsCalcola_Click(object sender, EventArgs e)
{
    Frazione FA = new Frazione ( Convert.ToInt16 ( txtNumA.Text ),
                                 Convert.ToInt16 ( txtDenA.Text ) );
```

… continuiamo con la creazione della frazioni FB e della frazione risultato FR:

```
    Frazione FB = new Frazione ( Convert.ToInt16 ( txtNumB.Text ),
                                 Convert.ToInt16 ( txtDenB.Text ) );
    Frazione FR = new Frazione ( );
```

La creazione di FR richiama un ***costruttore "vuoto"***, ossia privo di parametri. Questo perché FR è la frazione risultato: essa non viene inizializzata in base a degli input, bensì calcolata in seguito dal programma stesso.

Uso dei Metodi di Oggetti di una Nuova Classe

Generalmente, una Nuova Classe prevede dei **Metodi** ossia delle "operazioni" che gli oggetti della classe devono essere in grado di realizzare in modo autonomo. Spesso, per poter effettuare un'operazione, il *Metodo* ha bisogno di **uno o più Parametri**.

Un *Metodo* deve essere richiamato con la notazione

<*nome-istanza*> **.** <*metodo*> **(** <*parametri*> **)**

I parametri, nella chiamata ad un Metodo, possono essere *0, 1 o più di 1* e devono essere *separati da una virgola,* proprio come in una normale chiamata ad un sottoprogramma.

☞ Nell'esempio, se l'utente seleziona il RadioButton relativo all'operazione Addizione, devi richiamare il **metodo Somma** dell'istanza FR per "ordinare" ad FR di "calcolarsi" come somma di FA e FB (che sono i parametri):

```
if ( radAddizione.Checked )
    FR.Somma(FA, FB);
```

Se FA e FB sono 2/3 e 5/4, dopo l'esecuzione del metodo *FR.Somma(FA, FB)*, l'istanza FR contiene la frazione 23/12.

Procedi in modo analogo anche per le altre operazioni:

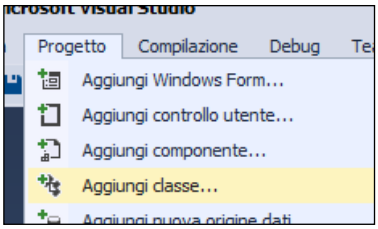
```
else if (radSottrazione.Checked )       FR. Differenza (FA, FB);
else if (radMoltiplicazione.Checked )   FR. Prodotto (FA, FB);
else                                    FR. Quoziente (FA, FB);
```

Infine, per visualizzare il risultato, richiami un ulteriore **metodo Visualizza** che, ricevendo come parametri due Label, provvede a visualizzare in esse il numeratore e il denominatore della frazione risultato:

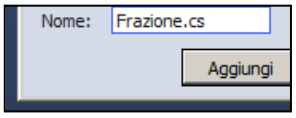
```
FR. Visualizza ( lblNum, lblDen );
```

Nel caso del *metodo Visualizza*, l'operazione implementata consiste nella visualizzazione (output) della frazione e i parametri previsti sono i due controlli Label in cui far apparire il numeratore e il denominatore della frazione.

Creare una Nuova Classe

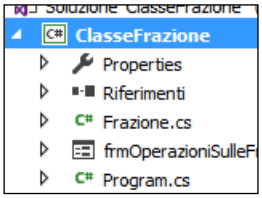


Per Creare un Nuova Classe, in ambiente Visual Studio, si utilizza il **menù Progetto / comando Aggiungi Classe.**



Si procede digitando il **Nome della Classe**.

Una *Classe*, al contrario di una *form*, è costituita **solo da una pagina di codice** e viene memorizzata in un file con estensione "**.cs**" (C Sharp).



☞ Nel nostro esempio creiamo una *Nuova Classe* con il *comando Progetto / Aggiungi Classe*, scegliamo il *modello Classe* e attribuiamo il nome *Frazione.cs*. Il nuovo elemento, ossia la *Classe Frazione*, appare nella finestra Esplora Soluzioni.

Il **Codice di una Classe** è così strutturato:

```
namespace  OperazioniSulleFrazioni      … nome del progetto …
{
    class Frazione                      … nome della classe …
    {
        ……                              … codice della classe …
    }
}
```

Struttura Dati interna e Attributi

Una Nuova Classe deve dichiarare al proprio interno una **Struttura Dati**, con le *Variabili* e gli *Oggetti Interni* (**Attributi**) necessari a **gestire i dati che caratterizzano l'oggetto** che si desidera realizzare.

☞ Nel nostro caso, l'oggetto da realizzare è molto semplice: una *frazione*. Quali sono i dati che caratterizzano una frazione? Ovviamente un Numeratore e un Denominatore! Dichiariamo allora due variabili *_Num* e *_Den* per memorizzarli:

```
class Frazione {
    private  int  _Num;
    private  int  _Den;
```

Generalmente, le variabili e gli oggetti che compongono la *Struttura Dati* interna della classe, vengono dichiarati con la **clausola private**, per impedire che, chi utilizza la classe, possa accedere direttamente alla struttura dati e *comprometterne integrità e coerenza*.

Il concetto di **Incapsulamento** prevede che la Classe *renda pubblico soltanto quello che è strettamente necessario* per utilizzare l'oggetto (proprietà, metodi, eventi, ecc.) e *"nasconda" rigorosamente tutto il resto*.

☞ E' bene chiarire che ogni volta che si crea una istanza della classe Frazione (con la clausola *new*) *viene stanziata in memoria una Struttura Dati diversa,* per cui ***ogni istanza ha una propria Struttura Dati*** indipendente dalle altre.

Metodo Costruttore

Per definire, in una classe C#, un **Costruttore con Parametri**, si implementa un particolare sottoprogramma avente lo **stesso nome della classe** e la **dichiarazione dei parametri** necessari alla creazione dell'oggetto.

☞ Nella nostra classe Frazione, implementiamo il metodo costruttore visto in precedenza: esso prevedeva due parametri, per stabilire il numeratore e il denominatore con cui inizializzare la frazione:

```
class Frazione {
    private  int  _Num;
    private  int  _Den;

    public  Frazione  (  int  nuovoNum,  int  nuovoDen  )
    {    _Num  =  nuovoNum;
         _Den  = nuovoDen;
    }
```

☞ In uno degli esempi visti precedentemente, creavamo una istanza con: **FA = new Frazione (4, 5) ;**
Questa istruzione provoca l'esecuzione del costruttore e trasmette i *valori 4 e 5* nei parametri *nuovoNum e nuovoDen*.

Compito del costruttore è "creare" e "inizializzare" l'oggetto, sulla base dei parametri che riceve: il codice del costruttore non fa altro che assegnare, agli attributi "interni" *_Num* e *_Den*, i valori dei rispettivi parametri *nuovoNum* e *nuovoDen*.

Alla fine otteniamo un'istanza FA di classe Frazione contenente la frazione "4 / 5".

Definire i Metodi in una Nuova Classe

Per **definire un Metodo** in una Classe, è sufficiente scrivere un normale sottoprogramma (di tipo *void* o *funzione*) e renderlo accessibile dall'esterno con la clausola **public**.

Nel **definire i Parametri** che il metodo riceve, è importante tener presente che un metodo *viene sempre richiamato da una istanza,* con la notazione ***<istanza>.<metodo>(<parametri>)***: di conseguenza, **l'oggetto che richiama il metodo può essere considerato a tutti gli effetti come un "parametro" del metodo**, anzi, spesso è *il parametro più importante* sul quale il metodo agisce.

☞ Consideriamo, ad esempio, un *metodo Semplifica*: se vuoi richiamarlo per semplificare una frazione FR, dovrai scrivere:

```
FR.Semplica()
```

Come vedi, *sembra che il metodo non abbia parametri*, ma in realtà agisce sull'oggetto che lo richiama, ossia FR.
FR in effetti *è come se fosse un vero e proprio parametro* e non ne servono altri!

☞ Chiarito questo, progettiamo assieme il *metodo Prodotto* e scriviamone il codice. Gli elementi su cui deve agire il metodo Prodotto sono 3: la prima frazione da moltiplicare, la seconda frazione da moltiplicare; la frazione in cui mettere la prodotto. Quando lo richiami, però, *uno di essi deve essere l'oggetto che richiama il metodo* ed è più logico che, a richiamare il metodo, sia la frazione in cui depositare il risultato:

```
FR.Prodotto (FA, FB)
```

Se impostiamo così il metodo, allora l'*oggetto che richiama il metodo* sarà la frazione in cui mettere il prodotto. Avrai notato che, in questo modo, i *parametri* del metodo sono solo 2 (e non 3!).

☞ Per definire il metodo, scriviamo, nel codice della classe, un *normale sottoprogramma void*, con la seguente testata:

```
public  void  Prodotto (Frazione F1, Frazione F2)
```

Nota come il metodo viene definito *PUBLIC* (e non *Private*) per renderlo "*visibile*" e "*accessibile*" dall'esterno della classe.

Nel codice di un Metodo, **quando si utilizza una Variabile Interna della classe**, si sta operando sulla Variabile Interna **dell'oggetto che ha richiamato il metodo**.

☞ Visto che il metodo viene richiamato dalla form così: **FR.Prodotto(FA, FB)** allora l'oggetto che richiama il metodo è **FR**.
Se nel metodo usi **_Num** e **_Den**, stai utilizzando **_Num e _Den di FR**, quindi stai impostando proprio la frazione risultato.
Se, invece, usi **F1._Num** o **F1._Den**, allora stai operando con **_Num e _Den di F1** (che corrisponde a FA) …

```
public  void  Prodotto (Frazione F1, Frazione F2)
{    _Num   =   F1._Num * F2._Num;
     _Den   =   F1._Den * F2._Den;
}
```

✋ Osservazione importante: Le variabili interne, _Num e _Den, sono private e, quindi, non possono essere utilizzate "al di fuori" della classe. Infatti, nel codice della form, ***non si può scrivere FA._Num***. Questa regola non vale "all'interno" della classe, per cui, un'istanza di classe Frazione, come F1, dichiarata e usata "dentro" la classe, consente l'accesso ai membri privati: ***dentro la classe, è consentito scrivere F1._Num***.

Eventuali **istanze di una Classe, dichiarate e usate "all'interno" del codice della classe** stessa, **consentono l'accesso anche ai Membri Privati** della Classe.

☞ Come già esposto prima, *all'interno del codice* della classe, l'istanza F1 *permette di scrivere F1._Num*.
D'altro canto, nella form, ossia *all'esterno della classe, non è consentito scrivere FA._Num*.