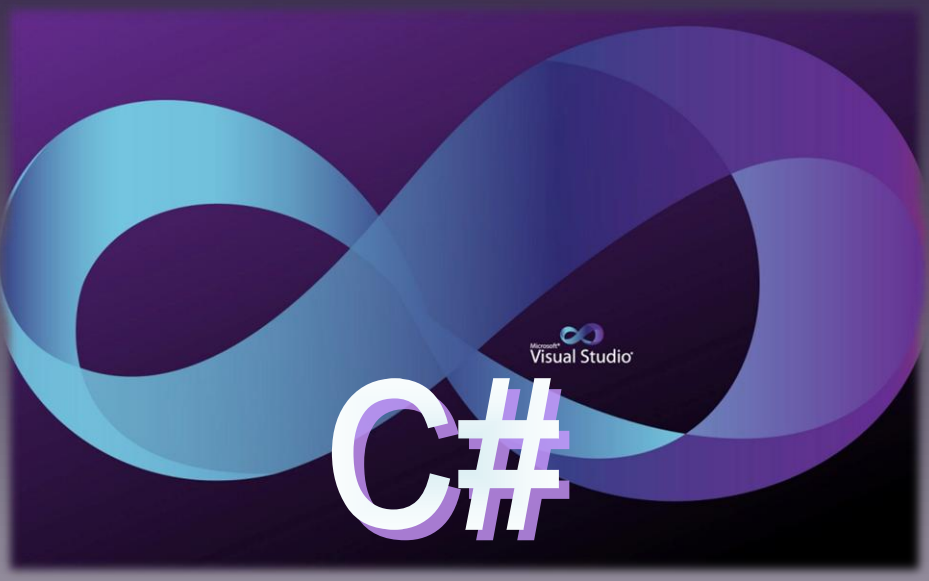
ISTITUTO TECNICO INDUSTRIALE
G. M. ANGIOY
SASSARI



CORSO DI PROGRAMMAZIONE

UTILIZZO DI PIÙ FINESTRE

DISPENSA 04.04

04-04_MultiForm_[ver_15]

Versione del: 07/11/2015
Revisione numero: 15

Prof. Andrea Zoccheddu
Dipartimento di Informatica

DIPARTIMENTO
INFORMATICA E TELECOMUNICAZIONI

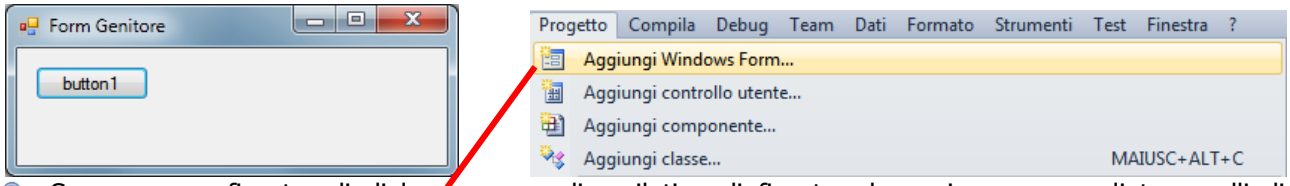
MULTIFORMS

FINESTRE MULTIPLE

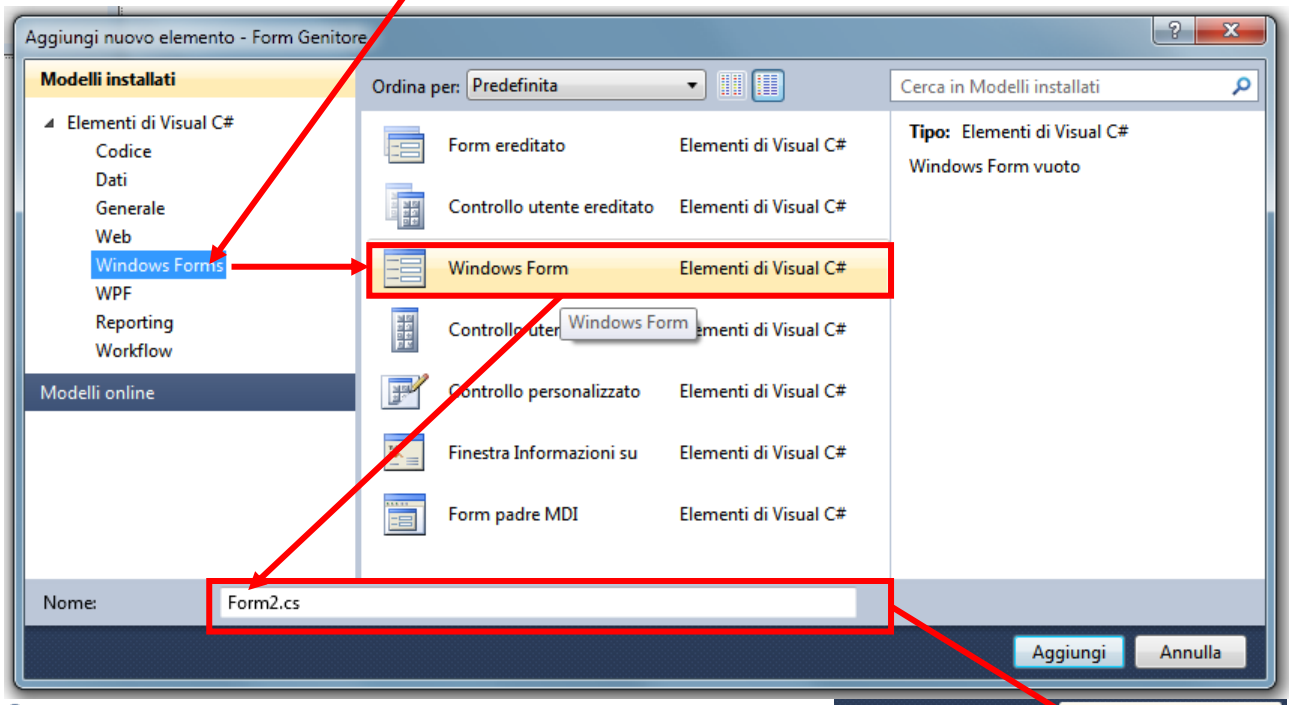
UNA SECONDA FINESTRA

PROGETTO GUIDATO

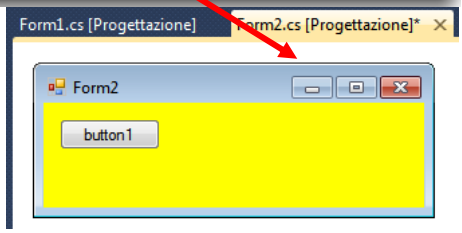
- Si prepari un Form1 simile alla figura a lato
- Dal menu Progetto si scelga la voce Aggiungi Windows Form . . .



- Compare una finestra di dialogo per scegliere il tipo di finestra da aggiungere; scegli tra quelli di `Windows Form` il tipo `Windows Form`



- Adesso ci sono due finestre nel progetto: osserva che sono riconoscibili dal titolo delle palette di progettazione.
- Inserisci un Button1 anche nel Form2
- Seleziona nuovamente il **`Form1`** (genitore) e fai doppio clic sul suo pulsante e scrivi il seguente codice:

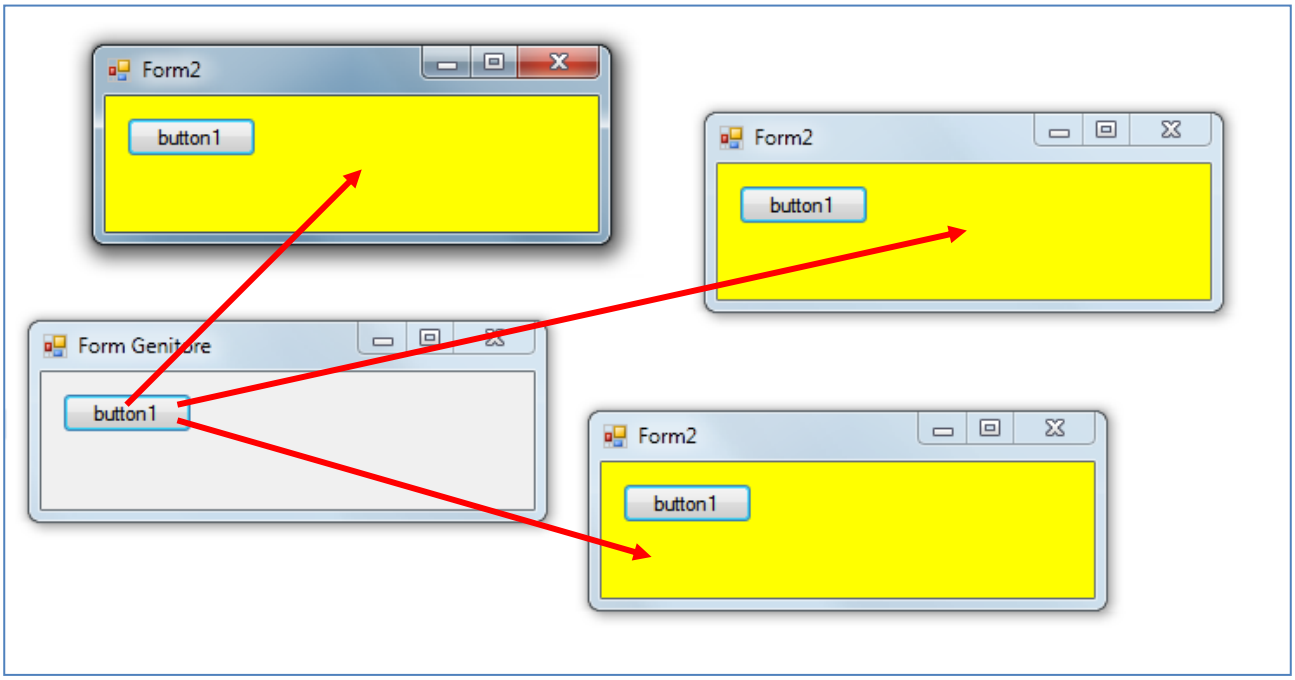



```
private void button1_Click(object sender, EventArgs e)
{
      //metodo del Form1 genitore
      Form2 figlia = new Form2();
      figlia.Show();
}
```

Seleziona nuovamente il **`Form2`** (figlio) e fai doppio clic sul suo pulsante e scrivi il seguente codice:

```
private void button1_Click(object sender, EventArgs e)
{
      //metodo del Form2 figlio
      MessageBox.Show("ciao");
}
```

- Prova il progetto. Hai notato che puoi generare tante finestre uguali?



- Torna alla fase di editing del progetto.
- Seleziona nuovamente il **Form1** (genitore) e fai doppio clic sul suo pulsante e modifica il codice come segue:

```
private void button1_Click(object sender, EventArgs e)
{
      //metodo del Form1 genitore
      Form2 figlia = new Form2();
      figlia.ShowDialog();
}
```

- Prova il progetto. Puoi ancora generare tante finestre uguali?

TANTE FINESTRE

LA FINESTRA COME TIPO DI DATO

Così come gli altri controlli, anche la finestra ha un suo tipo di dato e delle sue istanze. Per esempio nelle precedenti dispense abbiamo usato dichiarazioni come le seguenti:

```
Random dado = new Random();
Random tombola = new Random();
```

Nelle precedenti dichiarazioni la parola Random è il tipo, mentre la parola dado è una variabile. Quando si invoca la new specificando il nome del costruttore Random() si chiede di creare un'istanza.

Nella figura precedente si può osservare che:

- Il tipo determina la forma della variabile, ovvero costituisce il progetto secondo il quale la variabile è costruita;
- La variabile è un contenitore, che ospita valori concreti; nel caso di oggetti invece di ospitare l'oggetto vero e proprio, la variabile ospita un riferimento (un indirizzo o puntatore) per trovare l'oggetto costruito dal sistema;
- L'istanza è l'oggetto concreto; l'istanza è costruita invocando la new, indicando quale modalità deve si seguire per costruirlo correttamente.

Anche per le finestre si segue questo paradigma; quando si chiede al linguaggio di aggiungere una nuova finestra (un secondo Form), esso costruisce una dichiarazione che assume il seguente aspetto (è il codice in cui poi si scriveranno le istruzioni):

```
public partial class Form2 : Form
{
    public Form2()
    {
        InitializeComponent();
    }
}
```

Nella prima riga si dichiara un tipo di dato che ha il nome Form2. Questo tipo ha un blocco di parentesi graffe che racchiude tutto quello che serve.
La terza riga è un costruttore; ci interessa solo sapere che deve essere lasciato inalterato.

Per il momento non è importante sapere come funziona questa dichiarazione; il dettaglio di questi meccanismi sarà discusso in seguito, mentre per ora è sufficiente sapere che cosa si intende per Form2: il tipo su cui è costruita una finestra.

Quando si esegue un'istruzione come la seguente:

```
Form2 figlia = new Form2();
```

stiamo dichiarando una variabile di tipo Form2 e stiamo creando un'istanza.

non importa se i meccanismi non ti sono chiari, poiché per il momento è sufficiente che si abbia un'idea intuitiva del meccanismo di costruzione di finestre.

Come ultima considerazione, osserviamo che ogni volta che viene eseguita l'istruzione appena discussa (`Form2 figlia = new Form2();`) si crea una nuova istanza. Se il Form genitore è lasciato libero di eseguire più volte l'istruzione senza distruggere le finestre già create, appaiono tante finestre, ciascuna col lo stesso nome anche se in realtà sono variabili diverse; in pratica è come se avessi molte variabili che si chiamano tutte `figlia`.

FINESTRE MODALI E NON MODALI

Nei due esercizi guidati abbiamo usato due diversi metodi per far comparire la finestra:

- Show
- ShowDialog

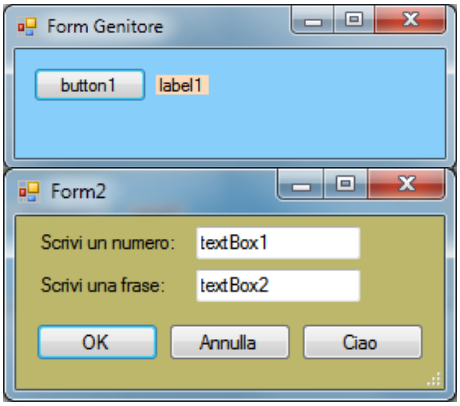
I due metodi hanno lo stesso effetto di mostrare sullo schermo la finestra ma hanno effetti diversi sul possessore della finestra, nel nostro caso il Form1.

Il metodo Show() in generale mostra un controllo a video; nel caso della fnestra essa compare nello schermo. il metodo non vincola la finestra genitore che quindi, dopo aver eseguito il metodo Show sulla variabile del figlio, può proseguire indipendentemente il suo lavoro.

Viceversa il metodo ShowDialog non è un metodo di qualsiasi controllo, ma è specifico per le finestre; il metodo mostra una finestra in modo Modale, ovvero in formato di dialogo che pone in attesa il genitore fino a quando la finestra figlia termina la sua esistenza (solitamente viene chiusa). Un dialogo modale impedisce di passare il focus al controllo padre (nel nostro caso Form1) e impedisce di poter interagire con esso fino alla conclusione del dialogo col figlio.

SCAMBIARE DATI

PROGETTO GUIDATO



- Si prepari un Form1 simile alla figura a lato
- Dal menu Progetto aggiunga un Form2 al progetto
- Si prepari il Form2 simile alla figura a lato
- Si visualizzi il codice relativo a Form2 (selezionando la paletta Form2.cs o col clic destro e visualizza codice)
- Si individui la dichiarazione della finestra Form2 e la dichiarazione del costruttore e si inserisca tra i due il seguente codice:

```
public partial class Form2
: Form
{
      //variabili della
finestra Form2
      public int esito = 0;
      public string frase =
"";
      //fine
      public Form2()
      {
      InitializeComponent();
      }
}
```

Inizio dichiarazione della finestra Form2

Devi scrivere qui, tra la dichiarazione della finestra Form2 e la dichiarazione del costruttore

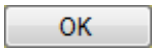
Qui dichiari delle variabili pubbliche (`public`)

Inizio dichiarazione del costruttore Form2

Del costruttore abbiamo accennato qualcosa, ma ti basti sapere che è quello invocato insieme alla new quando il genitore crea il figlio

- Dalla finestra Form2 scegli il button1 (`OK`) e col doppio-clic associa il codice seguente:

```
esito = 1;
frase = textBox1.Text;
Close();
```



- Dalla finestra Form2 scegli il button2 (`Annulla`) e col doppio-clic associa il codice seguente:

```
esito = 0;
Close();
```

- Dalla finestra Form2 scegli il button3 (`Ciao`) e col doppio-clic associa il codice seguente:

```
textBox1.Text = "";
esito = 2;
frase = "ciao";
MessageBox.Show("ciao!");
Close();
```

- Passa alla finestra Form1 e scegli il button1 e col doppio clic associa il codice seguente:

```
//pulsante del genitore
Form1
Form2 figlia = new
Form2();
figlia.ShowDialog();
if (figlia.esito == 0)
    label1.Text =
"Annullato";
if (figlia.esito == 1)
    label1.Text =
figlia.frase;
if (figlia.esito == 2)
    label1.Text +=
figlia.frase;
```

- Infine prova il progetto

DIALOGO PERSONALIZZATO

L'esercizio precedente mostra un semplice modo per scambiare dati tra la finestra figlia e la finestra genitore. Il meccanismo di base consiste nei seguenti passaggi:

1. Dichiarare variabili pubbliche (public) nella definizione della finestra figlia
2. Fare in modo che la finestra figlia modifichi le variabili pubbliche opportunamente a seconda della scelta dell'utente
3. Il genitore può impostare il valore delle variabili pubbliche prima di mostrarle con il metodo Show oppure con ShowDialog
4. Il genitore può usare il valore delle variabili pubbliche dopo aver concluso il lavoro con la finestra figlia

In questo modo è possibile far arrivare dati dal genitore al figlio e viceversa.

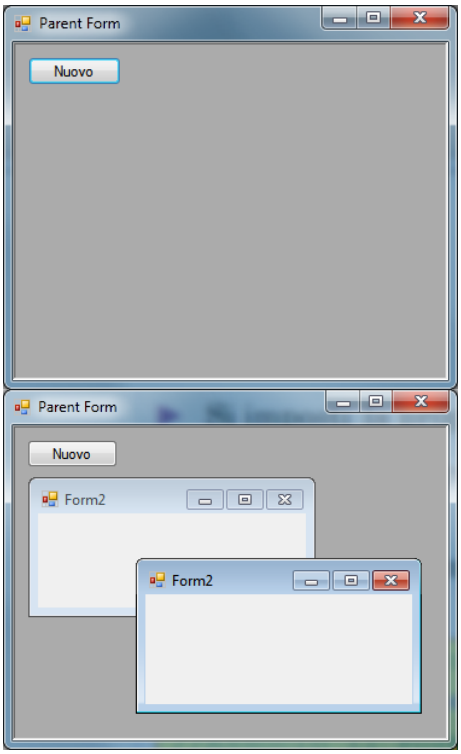
APPLICAZIONI MDI

PROGETTO GUIDATO

- Si prepari un Form1 simile al seguente:
- Si imposti la proprietà IsMdiContainer a True
- Dal menu Progetto aggiunga un Form2 al progetto
- Si torni al Form1 e si faccia doppio clic sul button1

```
//instanzio un nuovo Figlio
Form2 figlio = new Form2();
//Associo il Form Genitore
figlio.MdiParent = this;
//Mostro il figlio a video
figlio.Show();
```

- Si provi il progetto, usando qualche volta il button1
- Aggiustare il progetto abbellendo la Form2 a piacere



GENITORE E PROLE

Se lo scopo del programmatore è quello di sviluppare una applicazione dove ci sono più finestre incorporate all'interno di una unica finestra principale, diviene necessario utilizzare l'architettura MDI. In DOTNET è divenuto abbastanza semplice questa gestione multipla.

MDI significa Multiple Document Interface ovvero Interfaccia per Documenti Multipli; un'applicazione di tipo MDI permette di agire contemporaneamente con più documenti; per esempio possiamo aprire un'applicazione TextEditor che gestisce in parallelo più testi, ciascuno con una sua finestra.

Per implementare un simile progetto è indispensabile che vi sia un unico Form MDI principale detto Parent Form (Finestra Genitore); inoltre sarà possibile aprire più finestre dette Child Form (finestra Figlio) incorporate nel Form Genitore.

La proprietà `IsMdiContainer` del Parent Form deve essere `True`; questo implica che la Parent Form è pronta per accogliere altre finestre come sua prole. In automatico Visual Studio imposta lo sfondo del Parent Form a grigio, conferendo alla finestra quell'aspetto tipico delle finestre contenitori.

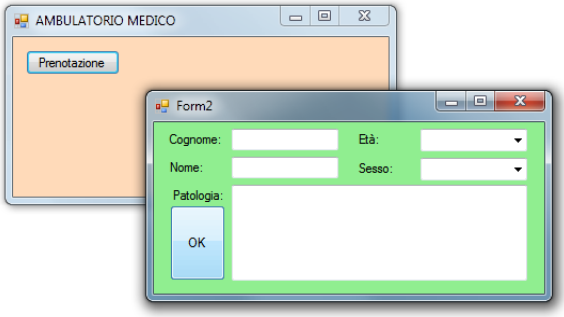
Quando affronteremo il discorso dei Menu, vedremo che, se in un Form Figlio si prevede la presenza di un menu diverso da quello principale, allora la barra dei menu cambierà automaticamente all'attivazione del Form Figlio.

ESERCIZI

ESERCIZI SUI FORM

Esercizio 1) DIALOGO DI INPUT

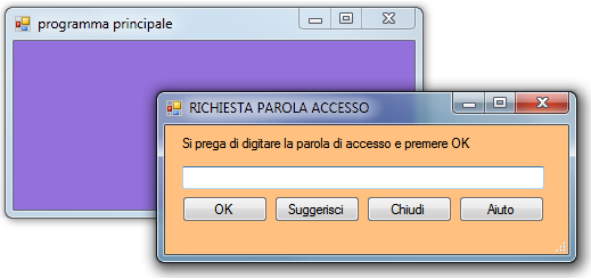
Si supponga che nell'ambulatorio di un medico, il paziente registri la sua richiesta; nella finestra principale c'è un pulsante che, quando premuto, propone una seconda finestra di dialogo con cui chiedere i dati del paziente ovvero Cognome e Nome, Sesso ed Età e patologia (es. malditesta). La finestra di dialogo usa un comboBox per l'età e un comboBox per il sesso, ed invece textBox per le altre informazioni. Il Form principale colleziona le informazioni in un vettore.



Esercizio 2) DIALOGO PASSWORD

Si vuole creare un'applicazione con una password di accesso.

Appena il Form principale appare, deve proporre un secondo Form per chiedere la password. L'utente può scegliere se digitare la password e premere OK oppure chiudere l'applicazione. Se l'utente sbaglia la password, il form principale la chiede subito nuovamente.



Arricchire l'applicazione proponendo un indovinello sulla password col pulsante suggerisci, chiudendo tutte le finestre con Chiudi e fornendo il numero di tentativi con Aiuto.

Esercizio 3) SCEGLI L'IMMAGINE

Si vuole creare un'applicazione che consente di scegliere un'immagine che potenzialmente) si userà per un gioco.

Alla pressione del pulsante Scegli appare una finestra di dialogo che consente di scegliere con un clic un'immagine che apparirà nella finestra principale, a fianco del pulsante.

Esercizio 4) COMPOSIZIONE DI PERIODI

Si vuole creare un'applicazione che consente di costruire una frase della forma:

<soggetto> <azione> <oggetto> <tempo>

Per esempio: *Il bimbo mangia la mela adesso.*

La finestra principale propone 4 pulsanti; alla pressione di un pulsante si modifica opportunamente la finestra Form2, che contiene un listBox1 vuoto che il Form1 riempirà come si deve.

La frase andrà componendosi nel riquadro azzurro del Form1.

SOMMARIO