Proiectarea aplicațiilor Windows în C#

Aplicațiile Windows sunt aplicații conduse (orientate) de evenimente.

Obiectul principal al unei aplicații Windows este formularul.

Spațiul **System.Windows.Forms** ne oferă clase specializate pentru crearea formularelor.

La crearea unei aplicații de tip **Windows Forms Application** sistemul generează un formular (fereastra principală a aplicației) ce va fi instanțiat(automat) în programul principal (metoda Main) printr-o secvență de cod ca mai jos:

```
static void Main()
{ ...
   Application.Run(new Form1());
}
```

Clasa *Application* este responsabilă cu administrarea unei aplicații Windows, punând la dispoziție proprietăți pentru a obține informații despre aplicație, metode de lucru cu aplicația și altele. Toate metodele și proprietățile clasei *Application* sunt statice. Metoda *Run* invocată mai sus creează un formular implicit, aplicația răspunzând la mesajele utilizatorului până când formularul va fi închis.

Putem adăuga proiectului noi formulare utilizând

Projects →Add Windows Form →Windows Forms →Templates Windows Form. In caseta *Name* atribuim o denumire noului formular.

După adăugarea unui nou formular, trebuie sa scriem si codul ce realizează instantierea și activarea lui.

Un formular poate fi activat cu metodele *Show*() sau *ShowDialog*(). Metoda *ShowDialog*() permite ca revenirea în fereastra din care a fost activat noul formular să se facă numai după ce noul formular a fost inchis (spunem că formularul nou este deschis modal).

Presupunând că am adăugat proiectului un formular nou cu denumirea formularNou, pentru instanțiere folosim o secvență de forma

formularNou f=new formularNou();
***f.* ShowDialog()**

Formularele (obiecte derivate din clasa **System.Windows.Forms.Form**) sunt obiecte de tip container(collection). Intr-un formular putem include **controale** adică obiecte(instanțe) de clase derivate din clasa **System.Windows.Forms.Control** care asigură funcționalitatea de bază a controalelor.

Proiectarea unui formular are la bază un cod complex, generat automat pe măsură ce adăugăm componentele și comportamentul acestuia.

Controale

Un *control* se caracterizează prin proprietățile sale și prin evenimentele care pot fi generate de el. Atunci când utilizatorul acționează asupra unui control(apăsarea unui buton, scrierea unui caracter, mișcarea mouse-ului etc.) se generează un

eveniment(*event*). Aplicația trebuie să sesizeze faptul că s-a acționat asupra controlului să identifice tipul acțiunii și să execute o secvență de cod care să rezolve acțiunea. Pentru aceasta, codul va trebui completat într-o metodă de tip *handler* (administrator de evenimente).

Un handler este o funcție specială care se lansează în execuție în mod automat la producerea evenimentului specific.

Metodele de tip handler au o sintaxă standardizată:

```
private void <denumire handler>_<tip eveniment>(object sender, System.EventArgs e)
  { //codul administratorului de eveniment }
```

Un handler primește ca parametri obiectul care produce acel eveniment(*sender*) și un obiect de tip *EventArgs* ce oferă detalii despre eveniment.

Marea majoritate a controalelor sunt obiecte de clase derivate din clasa ***System.Windows.Forms.Control***. Această clasă furnizează funcţionalitatea de bază pentru controale, de aceea multe din proprietăţile şi evenimentele controalelor, chiar dacă acestea sunt structural diferite, vor fi identice.

Proprietăţi ale controalelor

În cele ce urmează, vom prezenta un set de astfel de proprietăţi, furnizate de clasa ***Control***, deci comune tuturor controalelor. O descriere detaliată a acestor proprietăţi poate fi găsită în MSDN.

- **Anchor** – prin această proprietate se poate specifica comportamenul controlului la redimensionarea ferestrei ce-l conţine. Ancorarea lui de una din marginile ferestrei va face ca poziţia acestuia faţă de marginea respectivă să nu se schimbe. Ancorarea lui de toate marginile ferestrei va avea ca efect redimensionarea controlului odată cu redimensionarea ferestrei.

- **BackColor** – stabileşte culoarea de fond a controlului.

- **Bottom** – stabileşte distanţa de la marginea de sus a ferestrei la marginea de jos a controlului.

- **Dock** – ataşează controlul de una din marginile ferestrei.

- **Enabled** – activează controlul, adică permite acestuia să recepţioneze evenimente de la utilizator. Dacă această proprietate este FALSE, controlul este inactiv.

- **ForeColor** – defineşte culoarea textului.

- **Height** – defineşte înălţimea controlului.

- **Left** – defineşte marginea stângă a controlului, relativ la marginea stângă a ferestrei.

• **Name** – identificatorul controlului. Poate fi utilizat în cod pentru a individualiza şi accesa controlul.

• **Parent** – părintele controlului.

• **Right** - defineşte marginea dreaptă a controlului, relativ la marginea stângă a ferestrei.

• **TabIndex** – numărul asociat controlului în ordinea de tab. Ordinea de tab stabileşte ordinea in care controalele primesc ***input focusul*** (se selectează) la apăsarea tastei TAB.

• **TabStop** – Stabileşte dacă controlul primeşte input focusul la apăsarea tastei TAB. Dacă controlul nu are această proprietate, la apăsarea tastei TAB va fi sărit.

• **Tag** – este o proprietate care în general nu este utilizată de control. Poate fi doar un şir de caractere, care este stocat în interiorul controlului.

• **Top** - stabileşte distanţa de la marginea de sus a ferestrei la marginea de sus a controlului.

• **Visible** – dacă această proprietate este TRUE, controlul este vizibil. În caz contrar, deşi există în formular, el nu poate fi vazut.

• **Width** - stabileşte lăţimea controlului.

Evenimente generate de controale

Atunci când utilizatorul acţionează asupra unui control (apasă un buton, selectează o opţiune radio, scrie într-o listă, etc) aplicaţia trebuie să fie capabilă să sesizeze faptul ca s-a acţionat asupra controlului respectiv, să identifice tipul acţiunii şi să execute o secvenţă de cod care să rezolve problema în concordanţă cu tipul acţiunii. Pentru a sesiza acţiunea asupra lor, controalele generează evenimente. Clasa ***Control*** implementează un set de evenimente comune tuturor controalelor, iar acestea, la rândul lor, vor genera şi un set de evenimente specifice. Cele mai uzuale evenimente sunt prezentate mai jos, lista completă a acestora fiind disponibilă în MSDN.

• **Click** – este generat când se apasă click cu mouse-ul asupra unui control. Poate fi uneori generat şi de apăsarea tastei Enter.

• **DoubleClick** – este generat cînd se apasă dublu click asupra unui control. Dacă controlul este buton, acest eveniment nu va putea fi generat, pentru ca la prima apăsare se generează automat Click.

• **DragDrop** – este generat când se finalizează (este eliberat butonul mouse-lui) o operaţie drag-and-drop în care este implicat controlul.

• **DragEnter** – este generat atunci când obiectul implicat într-o operaţie drag-and-drop ajunge în interiorul controlului.

• **DragLeave** - este generat atunci când obiectul implicat într-o operaţie drag-and-drop părăseşte suprafaţa controlului.

• **DragOver** – este generat când un obiect implicat într-o operaţie drag-and-drop ajunge deasupra controlului.

• **KeyDown** – este generat când o tastă devine apăsată în timp ce controlul deţine input focusul. Se generează întotdeauna înainte de KeyPress şi KeyUp.

• **KeyPress** - este generat când o tastă devine apăsată în timp ce controlul deţine input focusul. Se generează întotdeauna după KeyDown şi înainte de KeyUp. Spre deosebire de KeyDown care furnizează codul de scanare al tastei apăsate, KeyPress furnizează codul ascii al tastei.

• **KeyUp** - este generat când o tastă este eliberată în timp ce controlul deţine input focusul. Se generează întotdeauna după KeyDown şi KeyPress.

• **GotFocus** – este generat când controlul primeşte input focusul.

• **LostFocus** – este generat când controlul pierde input focusul.

• **MouseDown** – este generat când prompterul mouse-lui este deasupra controlului şi se apasă o tastă a mouse-lui.

• **MouseMove** – este generat continuu atâta timp cât prompterul mouse-lui traversează controlul.

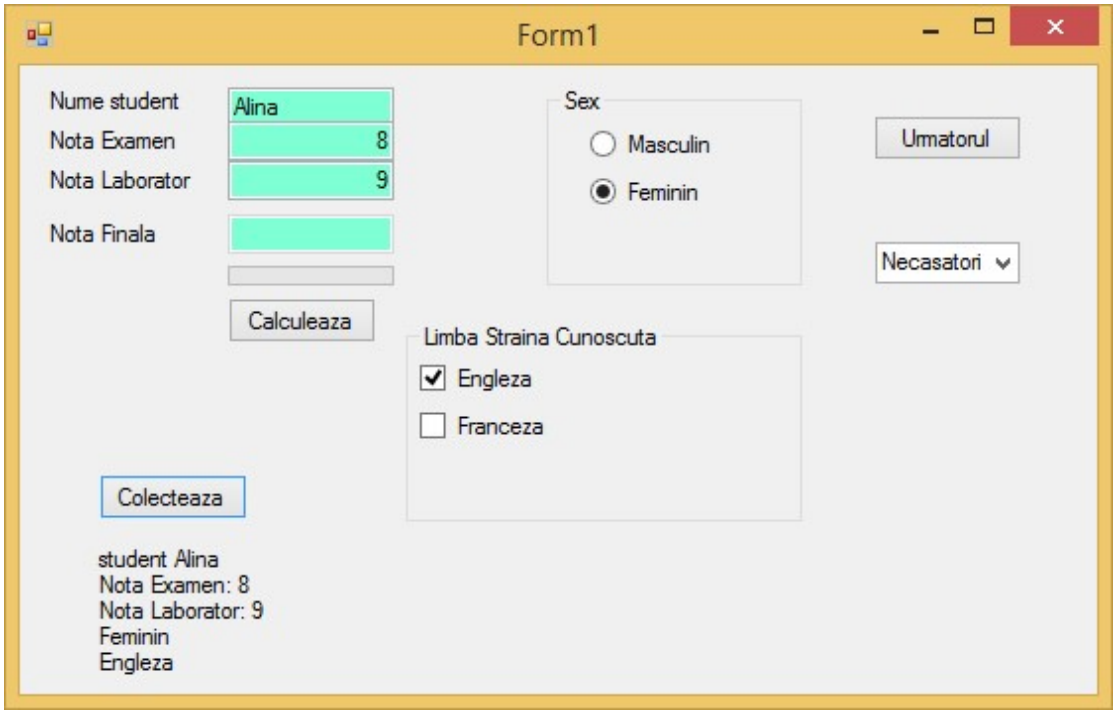
• **MouseUp** - ste generat când prompterul mouse-lui este deasupra controlului şi se eliberează o tastă a mouse-lui.

• **Paint** – se generează la desenarea controlului.

• **Validating** - este generat când un control este în curs de a primi focusul.

• **Validated** – este generat când un control este în curs de a primi focusul. Se generează după ce evenimentul Validating se termină şi indică faptul că validarea controlului este completă.
Evenimentele *Validating* si *Validated* sunt active doar daca proprietatea CausesValidation este setata cu *true*

Aplicatie




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Controale
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();

           cbStareCivila.Items.Add("Necasatorit");
           cbStareCivila.Items.AddRange(new
string[3]{"Casatorit","divortat","vaduv"});
           cbStareCivila.SelectedIndex = 0;
           progresMedie.Minimum = 0;
           progresMedie.Maximum = 100;

            foreach (object t in this.Controls)
                if (t.GetType() == typeof(TextBox))
                    ((TextBox)t).BackColor = Color.Aquamarine;
```

```
        }
        private bool NotaValida(Object sender)
        {
            TextBox t = (TextBox)sender;
            try
            {
                decimal n = decimal.Parse(t.Text);
                if (n < 5 || n > 10)
                    throw new Exception("Nota Eronata");
                return true;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Eroare:" + ex.Message);
                return false;
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            decimal medie;
            medie = (decimal.Parse(tbNotaEx.Text) +
decimal.Parse(tbNotaL.Text)) /2;
            tbNotaF.Text = medie.ToString();
            progresMedie.Value = (int)(progresMedie.Maximum * medie / 10);
        }

        private void tbNotaEx_Validated(object sender, EventArgs e)
        {
            if (!NotaValida((TextBox)sender))
                ((TextBox)sender).Focus();
        }

        private void tbNotaL_Validated(object sender, EventArgs e)
        {
            if (!NotaValida((TextBox)sender))
                ((TextBox)sender).Focus();
        }

        private void button2_Click(object sender, EventArgs e)
        {

            lbComentariu.Text = "student ";
            lbComentariu.Text += tbNumeStd.Text + "\n";
            lbComentariu.Text += "Nota Examen: " + tbNotaEx.Text + "\n";
            lbComentariu.Text += "Nota Laborator: " + tbNotaL.Text + "\n";
            if (rbFeminin.Checked) lbComentariu.Text += "Feminin" + "\n";
            else if (rbMasculin.Checked) lbComentariu.Text += "Masculin" +"\n";
            else lbComentariu.Text += "Nespecificat" + "\n";
            if (checkEN.Checked) lbComentariu.Text += "Engleza" + "\n";
            if (checkFR.Checked) lbComentariu.Text += "Franceza" + "\n";
        }

        private void button3_Click(object sender, EventArgs e)
        {

            tbNumeStd.Text = "";
            tbNotaL.Text = "";
            tbNotaEx.Text = "";
            tbNotaF.Text = "";
            lbComentariu.Text = "";
            checkFR.Checked = false;
```

```
            checkEN.Checked = false;
            rbFeminin.Checked = false;
            rbMasculin.Checked = false;
        }


    }
}
```