Introducere în Windows Forms

1. Introducere

Crearea unei aplicații Windows necesită utilizarea de clase din două assembly-uri standard:

- *System.Windows.Forms.dll*: conține clasele corespunzătoare elementelor grafice utilizate în cadrul aplicației (formulare, butoane, …) și mecanismul de execuție pentru acestea;
- *System.Drawing.dll*: conține structurile utilizate pentru reprezentarea geometriei (puncte, dreptunghiuri, dimensiuni, …) și funcțiile necesare pentru desenare pe ecran.

Pentru construirea unui proiect se vor urma următorii pași:

- se construiește un proiect nou de tip *Visual C# → Console Application*;
- se adaugă referințe la cele două assembly-uri menționate mai sus (click dreapta pe *References* în fereastra *Solution Explorer*, se alege *Add Reference…* și se selectează cele două assembly-uri din pagina *.NET*).

În cazul în care se dorește ascunderea ferestrei de consolă, se va modifica proprietatea *Output Type* din *Console Application* în *Windows Application* folosind fereastra de setări a proiectului.

2. Elemente de bază: clasele Control și Form

Un formular Windows Forms este reprezentat intern printr-un arbore de obiecte (vezi figura 1).

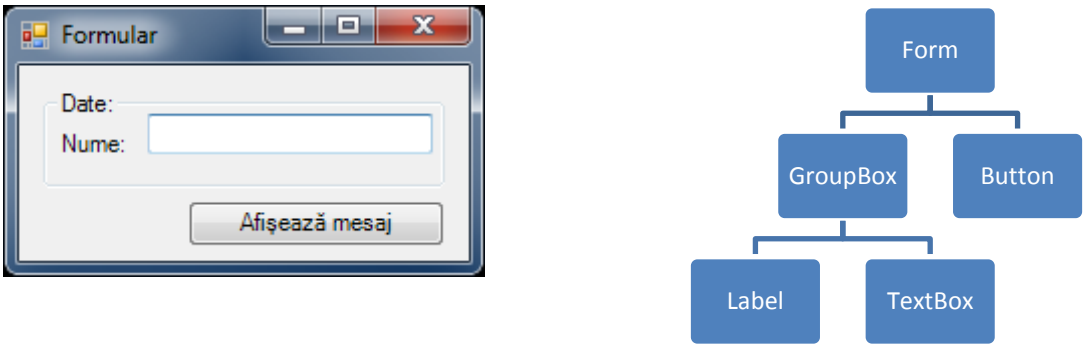

Figura 1: Arborele de controale asociat unui formular

Fiecărui obiect vizual prezent în cadrul unui formular (buton, câmp text, …) – denumit *control* - îi corespunde un obiect în cadrul arborelui. Toate aceste obiecte aparțin unei clase derivate direct sau indirect din clasa *System.Windows.Forms.Control* (vezi figura 2). Rădăcina arborelui trebuie să fie un obiect de tip System.Windows.Forms.Form (derivat și el din *System.Windows.Forms.Control*).

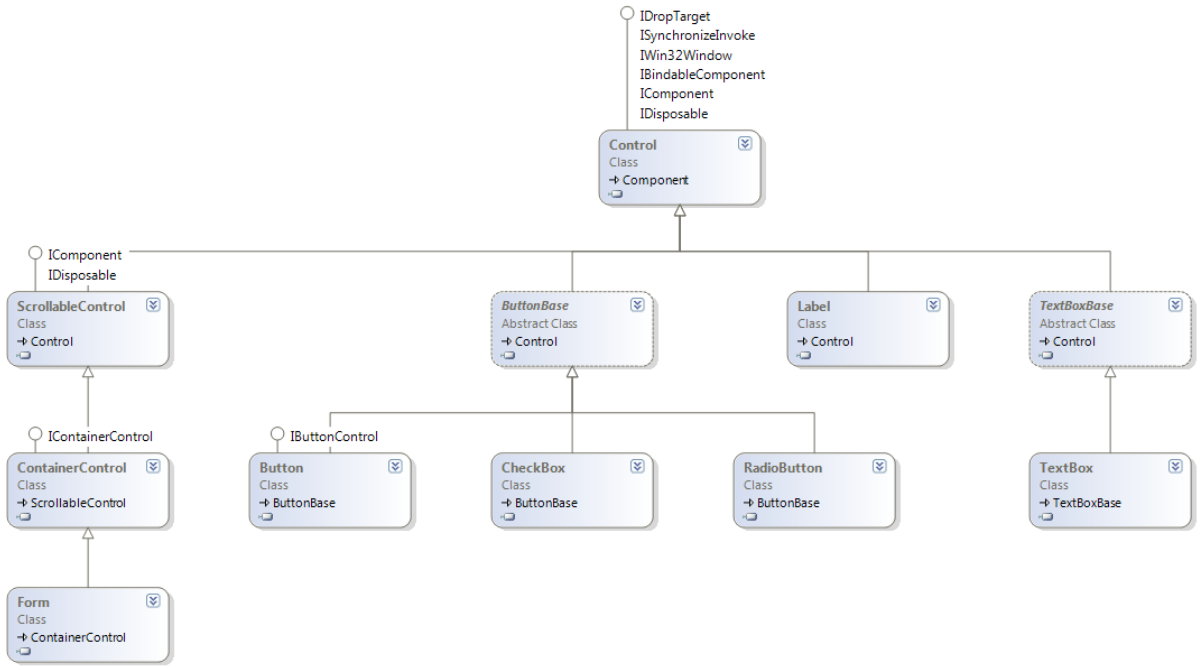

Figura 2: Diagrama de clase (simplificată)

Construirea unui formular funcțional presupune parcurgerea următoarelor etape:

- construirea obiectului formular și obiectelor corespunzătoare controalelor conținute;
- setarea proprietăților pentru fiecare obiect în parte (poziție, dimensiune, text, culori, …);
- adăugarea obiectelor în cadrul arborelui;
- furnizarea acțiunilor care trebuie executate de către controale (sub formă de obiecte *delegate*);
- lansarea formularului folosind metoda statică *Application.Run* sau una dintre metodele *ShowDialog* sau *Show* din clasa *Form*.

Exemplu 1: Construirea și utilizarea unui formular gol

```
using System;
using System.Windows.Forms;

class Program
{
    static void Main()
    {
        Form formular = new Form();
        formular.Text = "Test formular";

        Console.WriteLine("Inainte de afisare formular");
        Application.Run(formular);
        // apelul Application.Run se va incheia la inchiderea ferestrei;
        // dupa terminarea apelului obiectul 'formular' nu mai poate fi utilizat
        Console.WriteLine("Dupa afisare formular");
    }
}
```

Elementele necesare pentru construirea arborelui se regăsesc în cadrul clasei *Control*. Cele mai importante elemente sunt:

- *ControlCollection Controls {get;}*: proprietatea care conține legăturile către obiectele copil (derivate din *Control*); clasa *ControlCollection* conține metodele necesare pentru adăugarea, ștergerea și regăsirea controalelor copil;
- *Control Parent {get; set}*: conține o referință la controlul părinte (sau *null* în cazul rădăcinii arborelui); un control poate fi adăugat în cadrul arborelui prin setarea proprietății *Parent*;
- *Control TopLevelControl { get; }*: conține o referință la rădăcina arborelui.

3. Poziționarea controalelor

Fiecare control ocupă o zonă rectangulară în cadrul ferestrei. Punctul de origine al sistemului de coordonate este colțul din stânga sus al ferestrei. Unitatea de măsură utilizată este pixelul.

Pentru exprimarea elementelor geometrice se utilizează următoarele structuri de bază definite în System.Drawing:

- *Point*: reprezintă coordonatele unui punct (prin intermediul proprietăților X și Y);
- *Size*: reprezintă dimensiunile unui dreptunghi (prin intermediul proprietăților Width și Height).

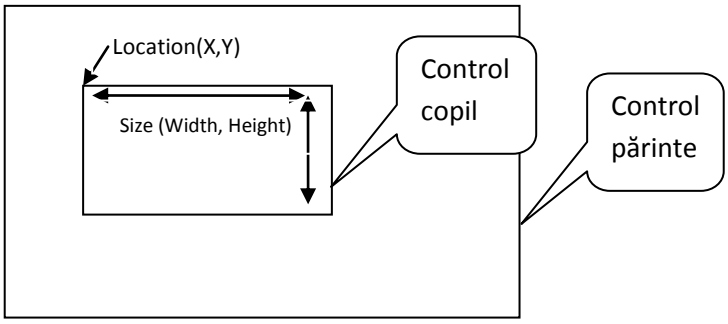

Figura 3: Poziționarea unui obiect Control

Cele mai importante proprietăți care determină zona ocupată de un control pe suprafața ferestrei sunt (figura 3):

- Location: proprietate de tip Point care determină poziția față de controlul părinte;
- Size: proprietate de tip Size care determină dimensiunea controlului.

Exemplu 2: Construirea formularului din figura 1

```
using System;
using System.Windows.Forms;
using System.Drawing;

class Program
{
    static void Main()
    {
        // 1. Construirea formularului
        Form formular = new Form();
        formular.Text = "Formular";
        formular.Size = new Size(250, 150);

        // 2. Construirea controalelor
        GroupBox group = new GroupBox();
        group.Location = new Point(12, 12);
        group.Size = new Size(200, 50);
        group.Text = "Date:";

        Label label = new Label();
        label.Location = new Point(7, 20);
        label.Size = new Size(40, 12);
        label.Text = "Nume:";

        TextBox text = new TextBox();
        text.Location = new Point(52, 12);
        text.Size = new Size(140, 20);
```

```
        Button button = new Button();
        button.Location = new Point(85, 70);
        button.Size = new Size(128, 23);
        button.Text = "Afișează mesaj";

        // 3. Construirea arborelui (legarea
controalelor)
        formular.Controls.Add(group);
        formular.Controls.Add(button);
        group.Controls.Add(label);
        group.Controls.Add(text);

        // 4. Afisarea formularului
        Application.Run(formular);
    }
}
```

În afara celor două proprietăți de bază menționate mai sus (*Location* și *Size*), clasa *Control* mai deține numeroase alte proprietăți care determină poziționarea controlului:

- *Top, Left*: permit citirea și modificarea distanței pe verticală și orizontală față de originea controlului părinte;
- *Bottom, Right*: permit citirea distanței de la marginea de jos / din dreapta a controlului față de controlul părinte;
- *Margin*: setează distanța dintre marginile controlului curent și controlul părinte;
- *Padding*: setează dintre marginile controlului curent și controalele copil;
- *Anchor*: determină modul în care proprietățile *Location* și *Size* ale controlului curent vor fi modificate automat la modificarea dimensiunilor controlului părinte;
- *Dock*: permite legarea dimensiunilor controlului curent de dimensiunile controlului părinte (vezi MSDN pentru detalii).

4. Alte proprietăți

Clasa Control conține o serie de proprietăți importante care sunt disponibile pentru toate controalele utilizate în cadrul bibliotecii *Windows Forms*. Modul concret în care sunt utilizate aceste proprietăți poate diferi de la un control la altul. De exemplu setarea proprietății *Text* va modifica titlul ferestrei pentru un obiect de tip *Form* sau textul afișat pentru un control de tip *Button*.

Cele mai importante proprietăți (în plus față de cele prezentate în secțiunile 2 și 3) sunt:

- *Text*: proprietate de tip *string* care permite modificarea textului afișat de către control;
- *Name*: proprietate de tip *string* care permite asocierea unei denumiri controlului pentru ușurarea regăsirii ulterioare în cadrul arborelui de controale;
- *Tag*: proprietate de tip *object* care permite asocierea unui obiect propriu controlului;
- *ForeColor, BackColor*: proprietăți de tip *System.Drawing.Color* care permit schimbarea culorii utilizate pentru textul și pentru fundalul controlului;
- *Font*: proprietate de tip *System.Drawing.Font* care permite modificarea stilului utilizat pentru afișarea textului în cadrul controlului;
- *Enabled*: proprietate de tip *bool* care permite activarea și dezactivarea controlului.

Exemplu – modificarea proprietăților controlului buton:

```
Button button = new Button();
...
button.BackColor = Color.Black; // fundal negru - culoare predefinita
button.ForeColor = Color.FromArgb(200, 0, 0); // text roșu - din componente RGB
button.Font = new Font("Arial", 8.5f, FontStyle.Bold);
button.Name = "butonAfisare";
```

5. Atașarea de acțiuni

Afișarea unui formular folosind *Application.Run* sau *ShowDialog* are ca efect declanșarea următorului proces (prezentat simplificat):

- Aplicația pornește o buclă în care preia mesajele primite de la sistemul de operare (apăsări de taste, modificări ale poziției mouse-ului, modificări ale ceasului de sistem, …);
- Pentru fiecare mesaj preluat se identifică controlul care este responsabil pentru procesarea mesajului și se apelează funcția corespunzătoare din controlul respectiv;
- Funcția apelată aplică modificările necesare asupra controlului (comută starea butonului, modifică textul afișat, …), după care apelează codul specificat de către utilizator (dacă există).

Specificarea codului utilizator care trebuie executat se realizează prin parcurgerea următorilor pași:

- Se identifică evenimentul pentru care dorim să atașăm acțiunea (exemplu: evenimentul *Click* generat la apăsarea mouse-ului deasupra controlului);
- Se obțin definițiile pentru eveniment și clasa delegat corespunzătoare din documentație sau din fereastra Object Explorer (exemplu pentru Click: *public event EventHandler Click* și *public delegate void EventHandler(object sender, EventArgs e);*);
- Se scrie o funcție de forma specificată în delegat (exemplu: *void FunctieButon(object sender, EventArgs e) {/*… cod de executat …*/}*);
- Se construiește un obiect delegat pe baza funcției definite anterior și se adaugă în lista de delegați a evenimentului (exemplu: *button.Click += new EventHandler(FunctieButon)*).

Clasele delegat definite pentru gestionarea evenimentelor în cadrul WindowsForms specifică două argumente:

- *sender*: de tip *object* care conține o referință la controlul care a declanșat evenimentul; acest parametru se utilizează pentru a determina sursa în cazul în care atașăm aceeași funcție la mai multe controale sau pentru a obține acces la celelalte controale din cadrul arborelui (folosind mecanismele prezentate în secțiunea 2);
- *e*: un obiect de tip *EventArgs* sau o clasă derivată din *EventArgs* care conține parametri suplimentari (tasta care a fost apăsată, poziția mouse-ului, …).

Exemplu 3: Atașarea de acțiuni

```
using System;
using System.Windows.Forms;
using System.Drawing;

class Program
{
    static void AfisareMesaj(object sender, EventArgs e)
    {
        // 1. Obținem o referință la controlul TextBox
        Button button = (Button)sender;
        Control formular = button.TopLevelControl;

        Control text = formular.Controls.Find("txtNume", true)[0];

        // 2. Construim și afișăm mesajul
        string mesaj = string.Format("Hello {0}!", text.Text);
        MessageBox.Show(mesaj);
    }
```

```
    static void Main()
    {
        // 1. Construirea formularului
        Form formular = new Form();
        formular.Text = "Formular";
        formular.Size = new Size(250, 150);

        // 2. Construirea controalelor
        GroupBox group = new GroupBox();
        group.Location = new Point(12, 12);
        group.Size = new Size(200, 50);
        group.Text = "Date:";

        Label label = new Label();
        label.Location = new Point(7, 20);
        label.Size = new Size(40, 12);
        label.Text = "Nume:";

        TextBox text = new TextBox();
        text.Location = new Point(52, 12);
        text.Size = new Size(140, 20);
        text.Name = "txtNume";   // atribuire nume pentru regăsire ulterioară

        Button button = new Button();
        button.Location = new Point(85, 70);
        button.Size = new Size(128, 23);
        button.Text = "Afișează mesaj";

        button.BackColor = Color.Black;
        button.ForeColor = Color.FromArgb(200, 0, 0);
        button.Font = new Font("Arial", 8.5f, FontStyle.Bold);

        // construire obiect delegate și adăugare în lista de acțiuni
        button.Click += new EventHandler(Program.AfisareMesaj);

        // 3. Construirea arborelui (legarea controalelor)
        formular.Controls.Add(group);
        formular.Controls.Add(button);
        group.Controls.Add(label);
        group.Controls.Add(text);

        // 4. Afisarea formularului
        Application.Run(formular);
    }
}
```

6. Controale simple

În cadrul bibliotecii Windows Forms sunt definite o serie de controale de bază:

- *Form*: clasa de bază pentru o fereastră; fiecare arbore de controale corespunzător unui formular trebuie să aibă exact un obiect de tip *Form* sau derivat din *Form* ca nod rădăcină;
- *Label*: permite afișarea unui text static (care nu poate fi modificat de către utilizator);
- *TextBox*: permite afișarea și citirea de text (texul poate fi modificat de către utilizator);
- *Button*: permite inițierea unei acțiuni prin apăsarea acestuia;
- *Panel*: control care nu afișează conținut dar permite gruparea mai multor controale copil;
- *CheckBox*: permite afișarea unui text static și modificarea de către utilizator a unei valori de tip *bool* (prin intermediul proprietății *Checked*);
- *GroupBox*: permite gruparea mai multor coloane (la fel ca *Panel*) și afișarea unui text;
- *RadioButton*: permite afișarea unui text static și modificarea de către utilizator a unei valori de tip *bool* (prin intermediul proprietății *Checked*); diferența față de CheckBox constă în

faptul că doar un singur control *radio* dintr-un grup (definit folosind *Panel* sau *GroupBox*) poate avea valoarea *true* la un moment de timp.

7. Structura standard pentru un formular

Pentru gestiunea mai facilă a formularelor se recomandă structurarea codului după următoarele reguli:

- Pentru fiecare formular se va defini o clasă derivată din *Form*;
- Pentru fiecare control prezent în arbore se va declara un câmp în interiorul clasei;
- Codul pentru crearea controalelor, setarea proprietăților și atașarea delegaților se va grupa într-o metodă denumită *InitializeComponent* care se va apela din constructorul clasei;
- Fiecare funcție de tratare a unui eveniment va declarată ca funcție membru în cadrul clasei (în acest fel va avea acces direct la toate controalele prin intermediul câmpurilor definite la pasul 2);
- Funcția Main va construi o instanță a clasei care reprezintă formularul principal al aplicației și va utiliza un apel Application.Run pentru pornire.

Exemplul 3: Restructurarea codului în formatul standard

```
using System;
using System.Windows.Forms;
using System.Drawing;

class FormularMesaj : Form
{
    Button button;
    GroupBox group;
    Label label;
    TextBox text;

    public FormularMesaj()
    {
        InitializeComponent();
    }

    void InitializeComponent()
    {
        this.Text = "Formular";
        this.Size = new Size(250, 150);

        group = new GroupBox();
        group.Location = new Point(12, 12);
        group.Size = new Size(200, 50);
        group.Text = "Date:";
        this.Controls.Add(group);

        label = new Label();
        label.Location = new Point(7, 20);
        label.Size = new Size(40, 12);
        label.Text = "Nume:";
        group.Controls.Add(label);

        text = new TextBox();
        text.Location = new Point(52, 12);
        text.Size = new Size(140, 20);
        text.Name = "txtNume";
        group.Controls.Add(text);

        button = new Button();
        button.Location = new Point(85, 70);
        button.Size = new Size(128, 23);
        button.Text = "Afișează mesaj";
```

```
        button.BackColor = Color.Black;
        button.ForeColor = Color.FromArgb(200, 0, 0);
        button.Font = new Font("Arial", 8.5f, FontStyle.Bold);

        button.Click += new EventHandler(this.AfisareMesaj);
        this.Controls.Add(button);
    }

    void AfisareMesaj(object sender, EventArgs e)
    {
        string mesaj = string.Format("Hello {0}!", text.Text);
        MessageBox.Show(mesaj);
    }
}

class Program
{
    static void Main()
    {
        FormularMesaj formular = new FormularMesaj();
        Application.Run(formular);
    }
}
```

Scrierea aplicațiilor în această structură este posibilă și prin utilizarea componentei Windows Forms Designer din Visual Studio. Funcționalitățile de bază oferite de Visual Studio pentru aplicații Windows Forms:

- Proiecte de tip Windows Forms Application care:
  - Au din start adăugate referințele pentru System.Windows.Forms și System.Drawing;
  - Este generat automat codul pentru programul principal și pentru un formular gol (structurat conform modelului de mai sus);
- La adăugarea unui formular nou:
  - Generează automat clasa derivată din *Form* cu structura de mai sus;
  - Codul corespunzător clasei este generat în două fișiere distincte:
    - Formular.cs: conține codul modificabil de către utilizator
    - Formular.Design.cs: conține codul generat automat și care nu poate fi modificat manual (câmpurile corespunzătoare controalelor, implementarea funcției InitializeComponent)
- La adăugarea unui control nou pe suprafața de design:
  - Adaugă un câmp nou în clasa formular corespunzător controlului adăugat;
  - Adaugă codul pentru crearea obiectului și legarea în cadrul arborelui în cadrul metodei InitializeComponent();
- La modificarea unei proprietăți: modifică / adaugă linia corespunzătoare în metoda InitializeComponent();
- La atașarea unei acțiuni (fereastra Properties, secțiunea Events):
  - Generează o metodă goală conformă cu definiția delegatului în fișierul Formular.cs
  - Generează codul pentru crearea obiectului delegat și adăugarea acestuia în listă în metoda InitializeComponent().

Pentru dezvoltarea unei aplicații se pot combina ambele metode (scrierea manuală de cod și utilizarea componentei designer). Utilizarea componentei designer nu este posibilă în cazul în care controalele trebuie construite la rulare.

Probleme:

1. Să se construiască o aplicație Windows Forms pentru calculul sumei a două numere cu zecimale.

2. Să se construiască un formular pentru citirea unui număr întreg cu un mesaj primit ca parametru. Să se folosească formularul pentru citirea unui vector de numere (pentru fiecare element se va afișa câte o fereastră) și să se afișeze suma lor.

   Indicații:

   - Se va construi un formular similar cu cel din exemplele de mai sus;
   - În clasa *Formular* se va adăuga o proprietate *Valoare* de tip întreg; funcția de *get* va folosi metoda *int.Parse* pentru a converti la întreg valoarea din textbox; funcția set va folosi metoda *ToString* pentru a converti valoarea primită și o va depune în textbox;
   - În programul principal, pentru fiecare număr, se vor executa următoarele:
     - Se va seta proprietatea Valoare pe 0;
     - Se va afișa formularul folosind metoda *ShowDialog*;
     - Se va adăuga în vector valoarea preluată din formular.Value.

3. Să se construiască un formular care să afișeze o tablă de șah. Numărul linii și coloane se va da ca parametru din funcția Main. La click pe oricare dintre celule se va afișa un mesaj în care se vor preciza coordonatele celulei (linia și coloana).

   Indicații:

   - Se va construi o clasă derivată din Form care va avea ca parametru numărul de linii / coloane - *n*;
   - Se vor genera folosind două construcții *for nxn* controale de tip *Panel*;
   - Pentru fiecare control Panel:
     - Se vor seta culoarea corespunzătoare (alb / negru), poziția și dimensiunea;
     - În proprietatea Tag se vor seta coordonatele pentru a putea fi utilizate ulterior la Click (de exemplu sub formă de vector de 2 întregi);
     - Se va atașa delegatul pentru tratarea evenimentului Click;
   - În funcția de tratare se vor extrage coordonatele din proprietatea Tag a controlului sursă și se vor afișa folosind MessageBox.Show.

4. Să se definească clasa Persoană (nume, sex – M/F). Să se construiască un formular pentru citirea unui obiect de tip Persoană. Formularul se va testa prin apelarea din Main și afișarea obiectului citit la consolă.
5. Să se construiască un formular cu un text static cu valoarea inițială “0” și două butoane cu textul “+”, respectiv “-”. La apăsarea unui buton valoarea afișată trebuie să crească sau să scadă cu o unitate (în funcție de butonul apăsat).