Laboratorul 1

Primii paşi în Visual Basic .NET

Ce ne propunem astăzi?

În laboratorul de astăzi ne propunem crearea unei aplicaţii simple pentru evidenţa studenţilor unei facultăţi. În cadrul acestei aplicaţii vom utiliza câteva dintre facilităţile POO oferite de Visual Basic .NET. Din păcate vom arăta numai o mică parte din acestea, urmând să învăţăm mai mult la curs şi pe parcursul lucrărilor de laborator ce vor urma.

Microsoft **Visual Studio.NET** este un mediu de dezvoltare folosit pentru crearea de aplicaţii în limbajele de programare: Visual Basic, C#, C++ sau J# (începând cu versiunea 2005).

Visual Studio.NET dispune de un editor complex de cod, care se foloseşte pentru crearea, modificarea şi depanarea codului aplicaţiei (poate fi vorba de cod Visual Basic .NET, Visual C# .NET etc.). De asemenea, oferă un set de obiecte şi instrumente cu ajutorul cărora se pot realiza cu uşurinţă interfeţe cu utilizatorul pentru Windows, Web, servicii WEB etc.

Visual Basic .NET este un limbaj de programare care face parte din familia de limbaje .NET, după cum o sugerează şi numele. Este un limbaj modern, puternic, complet orientat pe obiecte, la fel de valoros ca C# sau Java, care permite dezvoltarea atât de aplicaţii Windows cât şi de aplicaţii şi servicii WEB. O alta caracteristică demnă de amintit a acestui limbaj este aceea ca reuşeşte să păstreze sintaxa simplă care a fost întotdeauna elementul distinctiv al familiei de limbaje BASIC.

O aplicaţie tipică Windows, realizată în Visual Basic .NET, afişează unul sau mai multe ecrane conţinând obiecte cu care utilizatorul va interacţiona. Obiectele programului sunt ferestrele aplicaţiei (denumite de asemenea „Forms", sau formulare) şi controalele desenate pe ele în mod vizual.

În cadrul unei aplicaţii Windows formularele sunt elementele de bază ale interfeţei cu utilizatorul. În Visual Basic .NET formularele sunt complet orientate pe obiecte, ele reprezentând de fapt clase.

În momentul în care se creează un nou formular, fie deschizând un proiect nou de tip Windows Application, fie adăugând un nou formular proiectului existent (folosind caseta de dialog „Add New Item"), i se adaugă de fapt proiectului o clasă care derivă din clasa System.Windows.Forms.Form, căreia i se mai spune pe scurt clasa Form, iar noua clasă moşteneşte toţi membrii clasei existente.

Controalele sunt suportul interfeţei cu utilizatorul, al dialogului cu acesta. Acestea nu pot exista independent de formular, formularul fiind din acest punct de vedere un container cu mai multe controale.

După realizarea formularelor şi adăugarea controalelor dorite pe acestea, programatorul trebuie să adauge cod sursă şi să-l ataşeze unor evenimente care să permită utilizatorilor să

interacţioneze cu programul. De exemplu, dacă se doreşte ca un buton de comandă să deschidă un formular, în procedura corespunzătoare evenimentului Click asociată acelui buton, se va plasa codul corespunzător deschiderii unui nou formular.

Evenimentele se produc ca urmare a unei acţiuni a utilizatorului, a execuţiei programului sau pot fi declanşate de sistem. Producerea unui eveniment (Click, Double Click, Drag&Drop, KeyPress etc.) declanşează execuţia unei proceduri eveniment. Utilizatorul poate crea propriul cod în corpul acestei proceduri. Apelarea unei proceduri eveniment pentru un obiect dat se face automat, dacă obiectul e focalizat şi dacă se produce o acţiune care să declanşeze respectivul eveniment. Procedura eveniment poate fi apelată şi prin cod, la fel ca orice altă procedură, însă acest lucru nu va duce şi la declanşarea evenimentului asociat. Exemplu:

```
Nume_Obiect.Nume_Eveniment(Lista_Argumente)
```

Metodele obiectelor vizuale , spre deosebire de procedurile eveniment, sunt read-only. Ele pot fi doar apelate. Exista câteva metode comune tuturor obiectelor vizuale, în rest fiecare obiect având propriile lui metode. Metodele se apelează în felul următor:

```
Nume_Obiect.Nume_Metoda(Lista_Argumente)
```

Atât formularele cât şi controalele au asociate un set de proprietăţi prin intermediul cărora pot fi setate caracteristici ale acestora, cum ar fi:

- poziţia şi dimensiunea
- stilul şi comportamentul
- aspectele grafice etc.

Anumite proprietăţi pot fi modificate la proiectare (folosind fereastra de proprietăţi), altele pot fi modificate doar în timpul execuţiei, altele permit accesul şi la proiectare şi la execuţie. La execuţie, valoarea unei proprietăţi poate fi modificata prin instrucţiuni de forma:

```
Nume_Obiect.Nume_Proprietate = Valoare
```

Realizarea unei aplicaţii Visual Basic .NET înseamnă parcurgerea următoarelor etape:

- proiectarea mentală sau pe hârtie a interfeţei utilizator,
- crearea unui proiect nou,
- crearea de formulare (unul pentru fiecare fereastră a aplicaţiei),
- adăugarea de controale formularelor, folosind caseta de instrumente (ToolBox),
- crearea unei bare de meniu şi/sau a unei bare de instrumente (ToolBar) pentru funcţiile principale ale aplicaţiei (opţional),
- setarea proprietăţilor formularelor şi controalelor,
- scrierea codului (declaraţii de variabile, funcţii, proceduri sau clase, respectiv asocierea de cod pentru evenimentele obiectelor vizuale),
- testarea aplicaţiei,
- crearea fişierului executabil şi a kit-ului de distribuţie.

Noţiuni elementare de Programare Orientată pe Obiecte în Visual Basic .NET

Clase şi proprietăți

O clasă poate conţine câmpuri, metode sau proprietăţi. Membrii claselor pot avea oricare din specificatorii de acces permişi în Visual Basic: Private, Friend, Protected, Protected Friend şi Public.

Proprietăţile reprezintă modalităţi de expunere publică a câmpurilor unei clase, putând, la nevoie, să ofere numai drept de citire (read only) sau drept de scriere (write only). Avantajul utilizării proprietăţilor este acela că ele permit scrierea de cod asociat operaţiilor de atribuire şi verificarea în acest fel a validităţii datelor transmise.

În exemplul de mai jos se defineşte clasa Persoana care are două câmpuri private şi două proprietăţi publice care accesează câmpurile private *nume* şi *varsta*. Proprietatea *NumePersoana* poate fi accesată atât în mod citire cât şi în mod scriere, pe când proprietatea *VarstaPersoana* poate fi accesată numai în mod citire.

```
Class Persoana
    Private nume As String
    Private varsta As Byte

    Public Property NumePersoana() As String
        Get
            Return nume
        End Get
        Set(ByVal value As String)
            nume = value
        End Set
    End Property

    Public ReadOnly Property VarstaPersoana() As Byte
        Get
            Return varsta
        End Get
    End Property
End Class
```

Moştenire

Pentru a putea observa modul de utilizare a moştenirii în Microsoft Visual Basic, vom defini o clasă de bază:

```
Public Class Persoana
   ' campuri publice
   Public Nume As String
   Public Prenume As String

   Public Function NumeComplet() As String
      Return Nume & " " & Prenume
   End Function
End Class
```

Pentru a deriva o clasă Angajat din Persoana aveţi nevoie să adăugaţi o clauză Inherits imediat după declaraţia clasei:

```
' clasa Angajat moşteneşte clasa Persoana.
Public Class angajat
   Inherits Persoana

End Class
```

Interfețe

În linii mari, o interfaţă reprezintă o clasă abstractă, un set de membri pe care îi expune o clasă. De exemplu, toţi membrii publici ai unei clase se spune că aparţin interfeţei clasei, însă o clasă poate de asemenea să expună şi alte grupuri de proprietăţi şi metode care nu sunt vizibile public.

O interfaţă defineşte numai semnătura proprietăţilor şi metodelor (numele membrului, numărul şi tipul parametrilor, tipul şi valoarea returnată), pe când o clasă poate implementa acea interfaţă furnizând, după nevoie, codul pentru aceste proprietăţi şi metode. Codul din fiecare proprietate şi metodă poate să difere de la clasă la clasă, semantica fiecărei metode fiind păstrată. Faptul că fiecare clasă poate implementa aceeaşi proprietate sau metodă într-un mod diferit reprezintă baza pentru comportamentul polimorfic.

Puteţi defini o interfaţă în Visual Basic prin intermediul blocului Interface…End Interface:

```
Public Interface Interfata
    ReadOnly Property Denumire()
    Property Valoare()
    Sub Init()
End Interface
```

Interfeţele Visual Basic nu pot conţine cod executabil; puteţi include numai semnături de metode şi proprietăţi.

Iată cum ar arăta o clasă care implementează interfaţa definită mai sus:

```
Class Clasa
    Implements Interfata

    Public Sub Init() Implements Interfata.Init

    End Sub

    Public ReadOnly Property Denumire() As Object Implements Interfata.Denumire
        Get

        End Get
    End Property

    Public Property Valoare() As Object Implements Interfata.Valoare
        Get

        End Get
        Set(ByVal value As Object)

        End Set
    End Property
End Class
```

Partea practică a laboratorului

În cadrul acestui laborator se va crea în Visual Basic .NET o aplicaţie care să permită:

- crearea unei structuri de clase şi a unei liste de studenţi,
- afişarea într-un control de tip ListBox a studenţilor din cadrul unei facultăţi,
- adăugarea unui nou student,
- ordonarea studenţilor după nume,
- ordonarea studenţilor după medie descrescător,
- afişarea studenţilor dintr-un anumit an de studiu,
- căutarea unui student după nume,
- ştergerea unuia sau a mai multor studenţi (în funcţie de selecţia facută în ListBox).

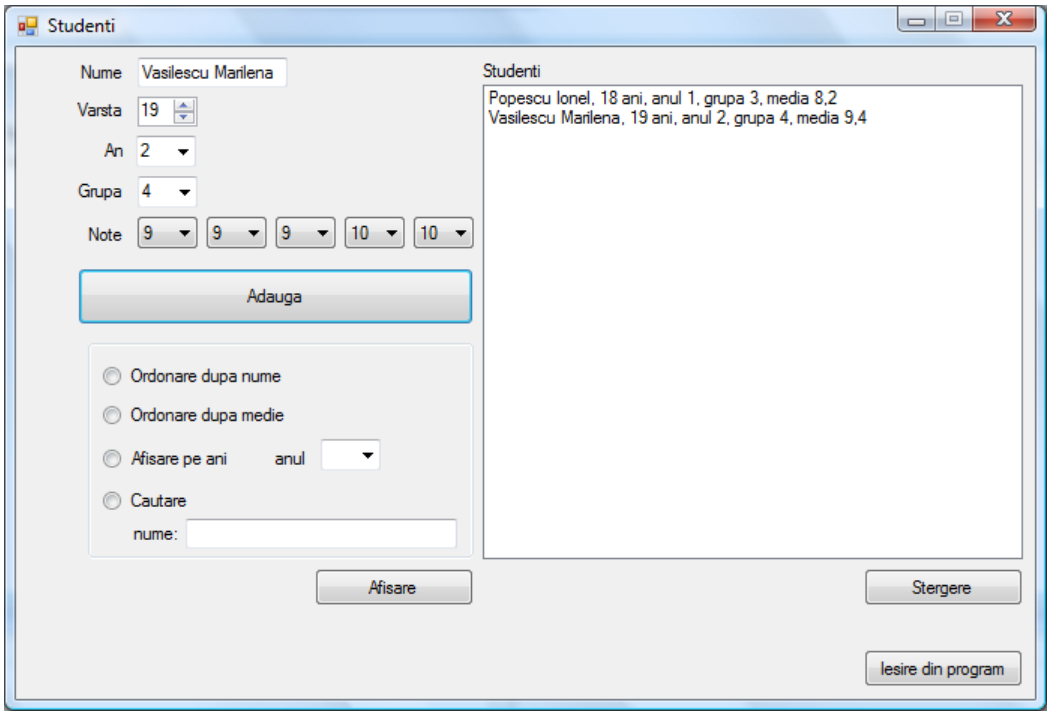

Figura 1. Fereastra aplicaţiei

Sfaturi utile

Nu pierdeți timpul încercând să construiți o interfață grafică extraordinar de spectaculoasă. Axați-vă în principal asupra aspectului funcțional al aplicației. Încercați să implementați toate cerințele aplicației, apoi testați aplicația şi eliminați posibilele erori!

Mod de lucru

Iată paşii care trebuie urmaţi:

- Se definesc doua clase: clasa Persoana (clasă de bază) şi clasa Student (clasă derivată).

- Clasa Persoana va avea ca membri:
  - Variabile protected: nume (String), varsta (Byte)
- Clasa Student va avea ca membri:
  - Variabile private: an (Byte), grupa (Byte), note(4) (Byte - şir de 5 note)
  - Proprietăţi publice read only: AnStudiu (Byte), NumeStudent (String)
  - Metode publice: Medie() As Single, AfisareStudent()

Observaţie:

- ✓ Este preferabil ca declararea celor două clase să se facă într-un modul nou de program (din meniul *Project*, comanda *Add Module*). De regulă modulele de program Visual Basic sunt utilizate atunci când dorim să declarăm clase, funcţii, variabile sau constante publice al căror domeniu de vizibilitate să fie extins la întregul proiect.

- Se implementează o listă de obiecte pentru clasa Student.

Exemplu:

```
Dim stud As New ArrayList   'crearea listei de studenti
Dim s as New Student        'crearea unui obiect de tip Student
'…
stud.Add s                  'adaugarea obiectului anterior creat in lista stud
CType(stud(0),Student).Afisare     'apelarea unei metode a primului obiect din lista
```

Observaţi că, pentru accesarea unui membru al listei, este nevoie de o conversie explicită a obiectului returnat de listă la tipul Student (funcţia CType din ultima linie de cod). Acest lucru se întâmplă deoarece membri listei sunt stocaţi ca obiecte de tip Object (acest tip de dată poate stoca orice tip de obiect).

- Se vor prevedea modalităţi de verificare a validităţii datelor introduse de la tastatură (cazul variabilelor numerice), prin utilizarea blocurilor Try…Catch.

Exemplu:

```
Try
    'instructiunile care pot provoca aparitia unei erori
Catch ex As Exception
    'Tratarea erorii, eventual prin afisarea descrierii erorii:
    'MessageBox.Show(ex.Message)
End Try
```

Observaţii:

- ✓ Proprietăţile read-only NumeStudent şi AnStudiu din clasa Student, dau acces în mod citire la datele studentului, care altfel nu ar fi fost accesibile din afara clasei.
- ✓ Sortarea studenţilor se va face prin metoda Sort a listei (ArrayList). Pentru aceasta trebuie definită o clasă care să implementeze interfaţa IComparer (aceasta are un singur membru: metoda publică Compare) şi să supraîncarce metoda Compare pentru a-i da sensul dorit.
  - Mai multe informaţii despre aceasta se pot găsi la o căutare în Help după: ArrayList.Sort Method (IComparer).

Cu ce ne-am ales?

Prin aplicația dezvoltată în cadrul laboratorului de astăzi am reuşit să ne familiarizăm pe de-o parte cu stilul de programare Visual Basic .NET, iar pe de altă parte am luat contact cu o parte din facilitățile POO oferite de acest limbaj.

Bibliografie

[1] http://msdn.microsoft.com/en-us/vbasic/default.aspx