TEHNICI DE PROGRAMARE AVANSATA – LABORATOR 8

1. Implementarea unui model de interactiune intre doua obiecte Form (Varianta 1)
2. Implementarea unui model de interactiune intre doua obiecte Form (Varianta 2)

1. Implementarea unui model de interactiune intre doua obiecte Form (varianta 1)

Sa consideram un form numit *form Principal* ce contine un comboBox cu anumite informatii obtinute dintr-o tabela T, a unei baze de date.

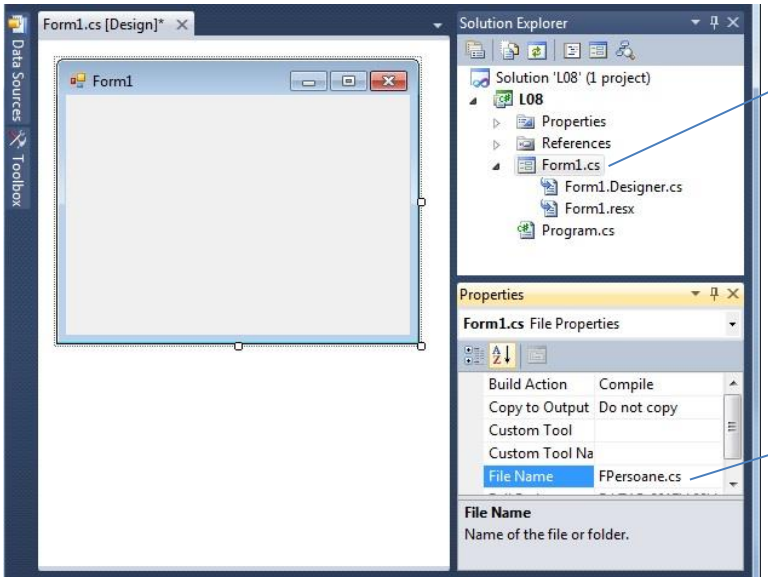
Sa consideram un al 2-lea form numit *form Secundar*, ce permite vizualizarea si actualizarea datelor din tabela T.

Form-ul principal poate solicita (de ex. prin apasarea unui buton) deschiderea form-ului secundar, care sa efectueze actualizari pe tabela T.

Dupa salvarea actualizarilor pe tabela T si inchiderea form-ului secundar se revine in form-ul principal. ComboBox-ul va prezenta prin deschidere, informatiile actualizate.

Observatie: comboBox-ul din form-ul principal poate fi inlocuit cu orice alt tip de colectie de date (grid, listBox, etc).

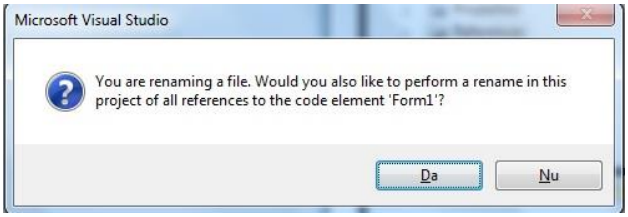
Exemplu. Vom crea o aplicatie noua. Schimbam numele form-ului Form1 in FPersoane:



1. Selectam Form1.cs

2. Schimbam numele form-ului: din Form1.cs in FPersoane.cs

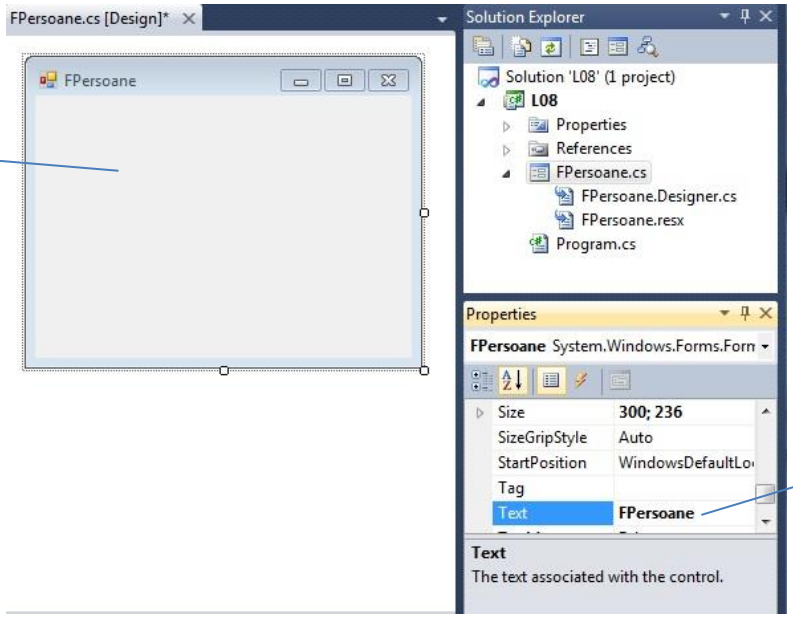
Sistemul ne afiseaza urmatoarea fereastra de dialog:



Confirmam redenumirea.

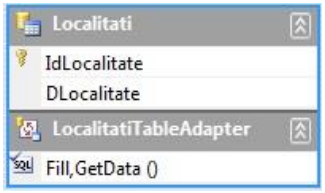
Schimbam apoi titlul form-ului.

1. Selectam form-ul



2. Schimbam propritatea Text

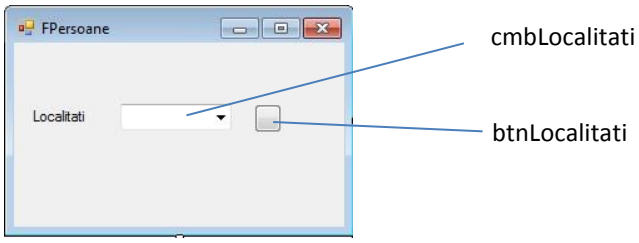
Aplicatia necesita un dataset care sa contina DataTable-ul Localitati.

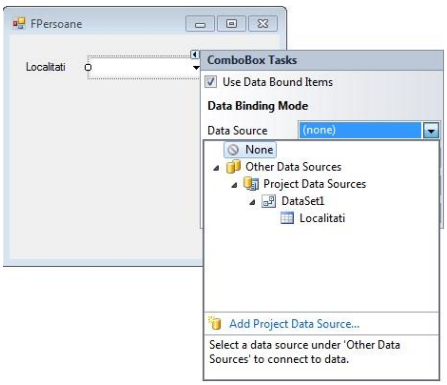


Query-ul pt. LocalitatiTableAdapter fiind:

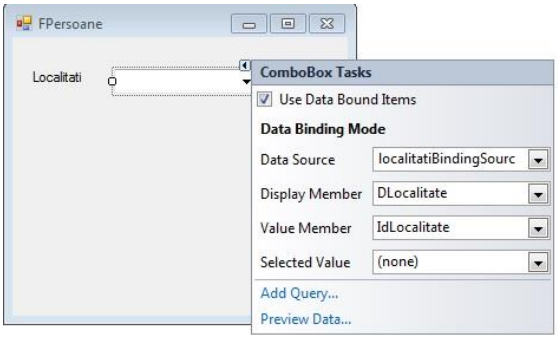
```
SELECT      IdLocalitate, DLocalitate
FROM        Localitati
ORDER BY DLocalitate
```

Sa consideram ca form principal form-ul FPersoane, ce permite vizualizarea si actualizarea de persoane, utilizand modelul simplu. Simplificand mai mult, retinem doar un comboBox (cmbLocalitati) din care putem selecta o localitate (localitatea de domiciliu) in timpul operatiei de adaugare sau modificare a unei persoane.



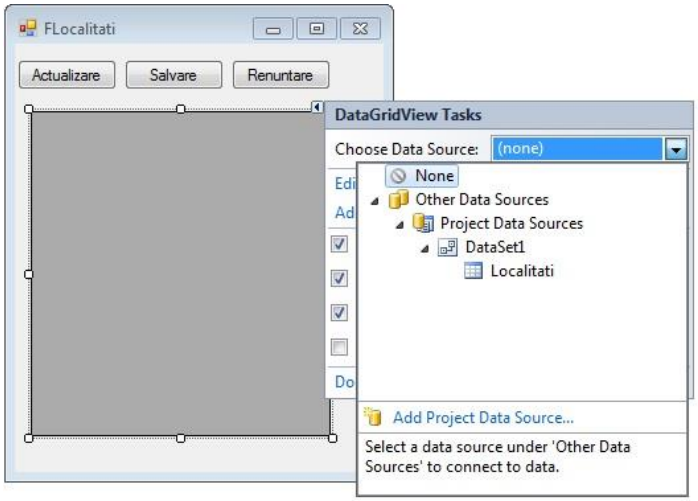


Configuram comboBox-ul:

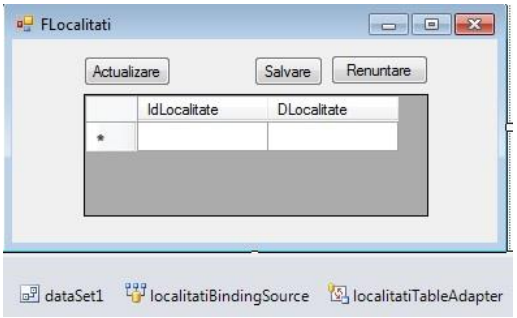


Sa consideram acum, form-ul FLocalitati (form secundar) care permite vizualizarea si actualizarea localitatilor. Putem utiliza modelul simplu de vizualizare/actualizare pe care-l vom simplifica si pe acesta (vom renunta la confirmarea stergerii si la validari).

Structura form-ului FLocalitati si legarea grid-ului la sursa de date:



Structura form-ului FLocalitati devine:



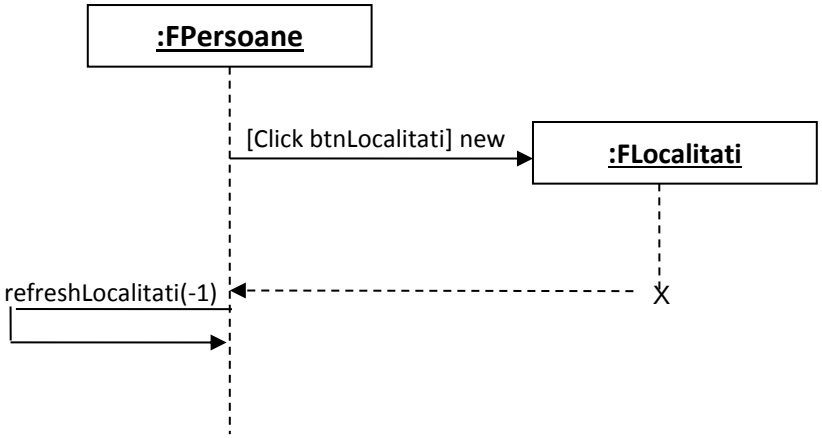
Modulul de cod (simplificat) pentru form-ul FLocalitati este:

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

namespace L08
{
    public partial class FLocalitati : Form
    {
        public FLocalitati()
        {
            InitializeComponent();
        }
```

```
        private void config(bool v) {
            dataGridView1.AllowUserToAddRows = !v;
            dataGridView1.AllowUserToDeleteRows = !v;
            dataGridView1.ReadOnly = v;

            btnActualizare.Enabled = v;

            btnSalvare.Visible = !v;
            btnRenuntare.Visible = !v;
        }

        private void refresh() {
            localitatiTableAdapter.Fill(dataSet1.Localitati);
        }

        private void FLocalitati_Load(object sender, EventArgs e) {
            config(true);
            refresh();
        }

        private void btnActualizare_Click(object sender, EventArgs e) {
            config(false);
        }

        private void btnSalvare_Click(object sender, EventArgs e){
            try {
                localitatiTableAdapter.Update(dataSet1.Localitati);
                config(true);
                refresh();
            }
            catch (Exception exc){
                MessageBox.Show(exc.Message);
            }
        }

        private void btnRenuntare_Click(object sender, EventArgs e){
            config(true);
            refresh();
        }
    }
}
```

Observatii:

1. Codul form-ului FLocalitati nu a suferit nici o modificare fata de modelul de vizualizare/actualizare standard.
2. Codul form-ului FLocalitati nu contine codul pentru validari si confirmari.
3. Form-ul de start este FPersoane.
4. Interactiunea dintre cele doua obiecte form poate fi descrisa printr-o diagrama de interctiune



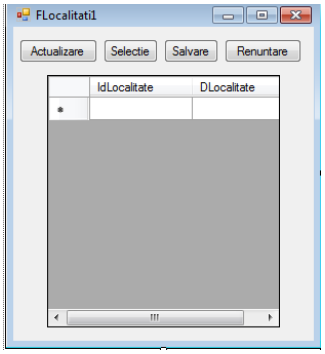
Modulul de cod pentru form-ul FPersoane este:

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

namespace L08
{
    public partial class FPersoane : Form
    {
        public FPersoane()
        {
            InitializeComponent();
        }

        private void refreshLocalitati(int pozitie){
            localitatiTableAdapter.Fill(dataSet1.Localitati);
            cmbLocalitati.SelectedIndex=pozitie;
        }

        private void FPersoane_Load(object sender, EventArgs e)
        {
            refreshLocalitati(-1);

        }

        private void btnLocalitati_Click(object sender, EventArgs e)
        {
            FLocalitati f = new FLocalitati();
            f.ShowDialog();
            refreshLocalitati(-1);
        }
    }
}
```

2. Implementarea unui model de interactiune intre doua obiecte Form (varianta 2)

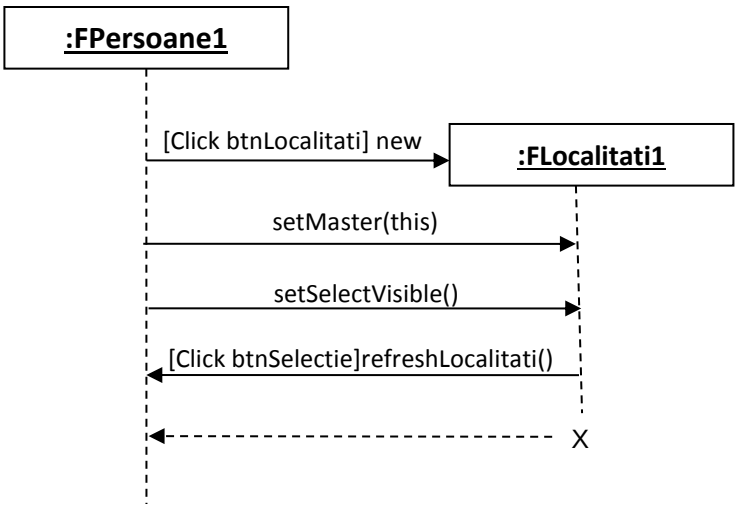
In aceasta varianta dorim ca la revenirea in form-ul principal sa gasim completata in comboBox, valoarea selectata in form-ul secundar. Astfel castigam timp la revenire in form-ul principal, sa nu mai deschidem comboBox-ul si sa cautam de exemplu, noua valoare introdusa. Vom crea doua noi form-uri: FPersoane1 si FLocalitati1.

Vom introduce un buton nou *btnSelectie* in form-ul FLocalitati1. Rolul acestui buton este de a transfera localitatea selectata in form-ul principal (FPersoane1) si de a inchide form-ul FLocalitati1. Avem insa o problema: form-ul FLocalitati1 poate fi utilizat si de sine statator, nu dintr-o intaractiune cu un alt form, caz in care butonul de selectie nu ar avea sens. Asa incat, implicit butonul btnSelectie va fi invizibil si doar in cazul unei interactiuni va deveni vizibil. El va trebui sa intre in mecanismul de disponibilitate specific logicii modelului respectiv.



Avem deasemenea nevoie de o variabila *master* care, in momentul declansarii interactiunii, va contine o referinta catre form-ul principal, care initiaza interactiunea.

Diagrama de interactiune ar putea fi :



Codul sursa al form-ului FPersoane1 are modificata doar metoda btnLocalitati_Click:

```
        private void btnLocalitati_Click(object sender, EventArgs e)
        {
            FLocalitati f = new FLocalitati();
            f.SetMaster(this);
            f.SetSelectVisible();
            f.ShowDialog();
        }
```

Codul sursa al form-ului FLocalitati1 este prezentat integral:

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

namespace L08
{
    public partial class FLocalitati1 : Form
    {
        private Form master;
        private bool selectie;

        public FLocalitati1()
        {
            InitializeComponent();
        }

        public void SetMaster(Form caller)
        {
            master = caller;
        }
```

```
public void SetSelectVisible()
{
    selectie  = true;
}

private void config(bool v)
{
    dataGridView1.AllowUserToAddRows = !v;
    dataGridView1.AllowUserToDeleteRows = !v;
    dataGridView1.ReadOnly = v;

    btnActualizare.Enabled = v;

    btnSalvare.Visible = !v;
    btnRenuntare.Visible = !v;

    if (selectie) btnSelectie.Visible = v;
}

private void refresh()
{
    int pozitie;
    pozitie = localitatiBindingSource.Position;
    localitatiTableAdapter.Fill(dataSet1.Localitati);
    localitatiBindingSource.Position = pozitie;
}

private void btnActualizare_Click(object sender, EventArgs e)
{
    config(false);
}

private void btnSalvare_Click(object sender, EventArgs e)
{
    try
    {
        localitatiTableAdapter.Update(dataSet1.Localitati);
        config(true);
        refresh();
    }
    catch (Exception exc)
    {
        MessageBox.Show(exc.Message);
    }
}

private void btnRenuntare_Click(object sender, EventArgs e)
{
    config(true);
    refresh();
}

private void FLocalitati1_Load(object sender, EventArgs e)
{
    btnSelectie.Visible = false;
    config(true);
    refresh();
}

private void btnSelectie_Click(object sender, EventArgs e)
{
    FPersoane1 f = (FPersoane1)master;
    f.refreshLocalitati(localitatiBindingSource.Position);
    this.Close();
```

```
        }
    }
}
```

Observatii: 1. Metoda refreshLocalitati trebuie declarata publica (in form-ul principal FPersoane1)
2. Form-ul FLocalitati1 ar trebui testat atat ca form-independent cat si prin interactiunea cu form-ul FPersoane1.