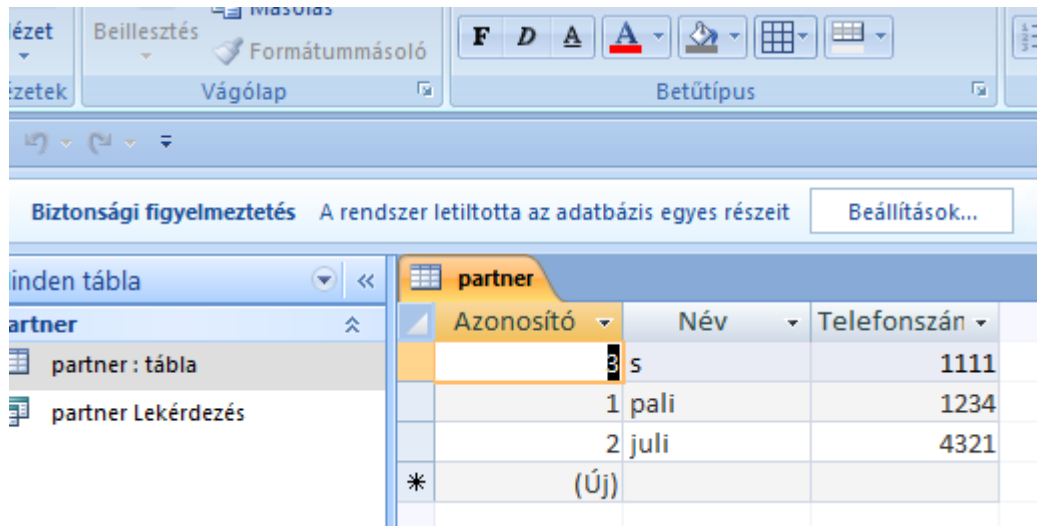
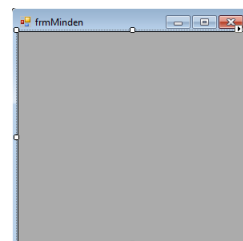
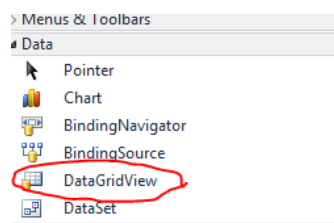


C#---Access adatbáziskezelési gyakorlat

1. Feladat: adatbázis kezelő alkalmazás készítése C# környezetben, Access adatbázist használva.
2. Minta adatbázis létrehozása ACCESS-ben



3. A Projekt létrehozása
4. Database kapcsolat létrehozása `date/addnewdatasource/database/dataset/New connection`:
 - a. Data source: Change/Microsoft Access Data File
 - b. Browse/adatbázsi elérési útja...
 - c. Test Connection
 - d. Next/Igen: bemásolja az adatbázist a projektbe, így könnyen szállítható, kompakt alkalmazást kapunk
 - e. Connection stringet másoljuk egy jegyzettömb fájlba és mentjük
 - f. Choose Your Database Objects: Tables, Views
 - g. Finish
5. Menü: Kilépés, Lekérdezések (minden, ...), Törlés, módosítás (Törlés, Módosítás), Adatbevitel
 - a. Kilépés: `Application.Exit()`;
6. Új form (frmMinden) létrehozása a „Minden” lekérdezés számára
7. Rács elhelyezése a form-on



8. A frmMinden formon egy kívülről írható tulajdonságot hozunk létre:

```
public partial class frmMinden : Form
{
    public frmMinden()
    {
        InitializeComponent();
    }
    public DataTable AdatForras
    {
        set
        {
            dataGridView1.DataSource = value;
        }
    }
}
```

9. `using System.Data.OleDb;` beállítása a fő formon

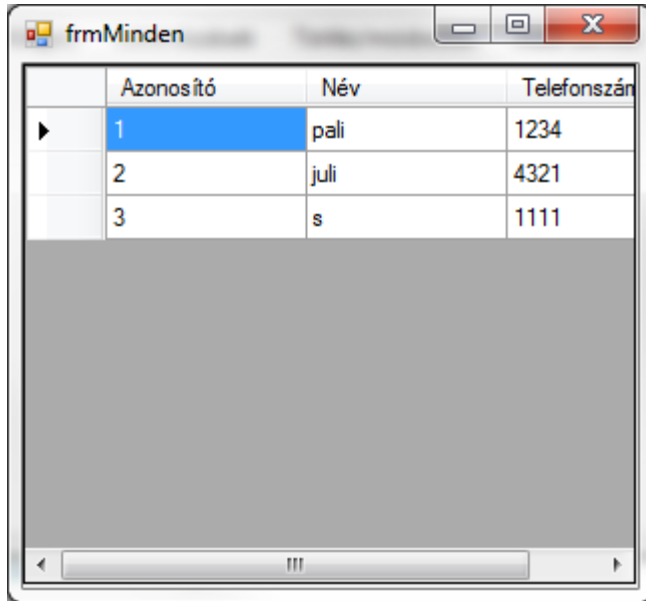
10. A „connection string”-t helyezzük egy globális változóba:

```
static string connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=|DataDirectory|\partnerek.accdb";
```

11. Eseménykezelő létrehozása(Click) a fő form, „Minden” menüponthoz

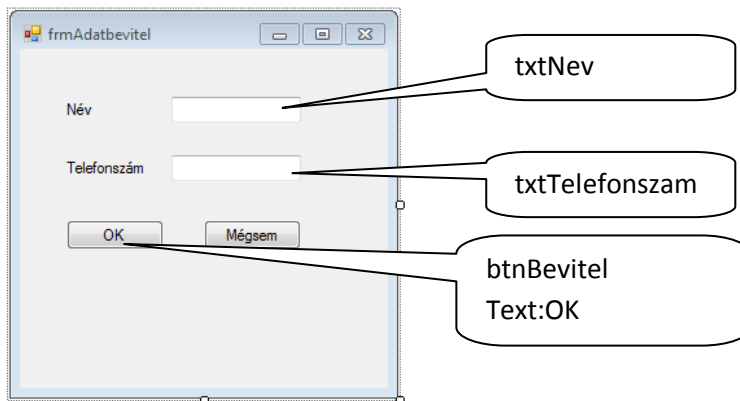
```
private void mindenToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmMinden form = new frmMinden(); //form létrehozása, még nincs megjelenítve
    OleDbConnection kapcsolat = new OleDbConnection();//adatbázis kapcsolat objektum
    kapcsolat.ConnectionString = connectionString;//kapcsolati string definiálása
    try
    {
        kapcsolat.Open();//kapcsolat megnyitása
        OleDbDataAdapter Adapter = new OleDbDataAdapter();//data adapter obj létrehozása
        Adapter.SelectCommand = new OleDbCommand("SELECT * FROM partner ", kapcsolat);
        DataSet dataset = new DataSet();//dataset obj létrehozása (ez a RAM-ban tárolja)
        Adapter.Fill(dataset);//dataset feltöltése az adapterrel
        form.AdatForras = dataset.Tables["Table"];// frmMinden formon lévő rács adatforrás
        kapcsolat.Close();
        form.Show();
    }
    catch (Exception kivetel)
    {
        MessageBox.Show(kivetel.Message, "Adatbázis hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

12. Lekérdezés ellenőrzése:



	Azonosító	Név	Telefonszám
▶	1	pali	1234
	2	juli	4321
	3	s	1111

13. Adatbevitel: frmAdatbevitel form



txtNev

txtTelefonszam

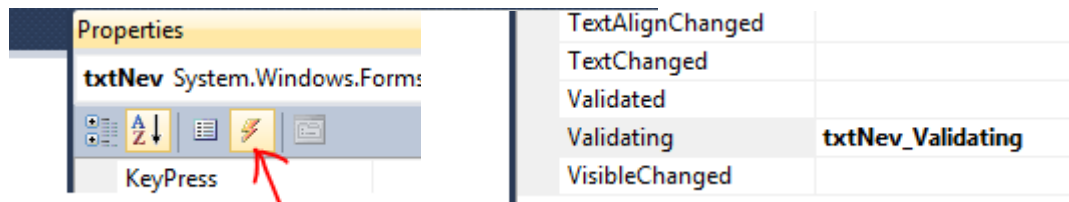
btnBevitel
Text:OK

14. `using System.Text.RegularExpressions;` //bevitel ellenőrzéséhez kell

15. Írható olvasható tulajdonságok létrehozása

```
public string Nev
{
    get { return txtNev.Text; }
    set { txtNev.Text = value; }
} //beállítható és lekérdezhető a mező a formon kívülről is
public string Telefonszam
{
    get { return txtTelefonszam.Text; }
    set { txtTelefonszam.Text = value; }
} //beállítható és lekérdezhető a mező a formon kívülről is
```

16. Esemény kezelőket írunk a bevitt adatok ellenőrzésére



```
,
// eseménykezelő ami csak akkor engedi elhagyni a mezőt ha a mezőben betű \
private void txtNev_Validating(object sender, CancelEventArgs e)
{
    string nev = txtNev.Text;
    for (int i = 0; i < nev.Length; i++)
        if (!Char.IsLetter(nev[i]) && nev[i] != ' ')
        {
            e.Cancel = true;
            MessageBox.Show("A névben csak betű és szóköz állhat!",
                "Adatbeviteli hiba", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            break;
        }
}

// eseménykezelő ami csak akkor engedi elhagyni a mezőt ha a mezőben szám \
private void txtTelefonszam_Validating(object sender, CancelEventArgs e)
{
    string telefon = txtTelefonszam.Text;
    for (int i = 0; i < telefon.Length; i++)
        if (!Char.IsDigit(telefon[i]) && telefon[i] != ' ')
        {
            e.Cancel = true;
            MessageBox.Show("A névben csak szám és szóköz állhat!",
                "Adatbeviteli hiba", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            break;
        }
}
```

17. Eseménykezelőt írunk az „OK” gomb click eseményéhez ami ellenőrzi, hogy valamelyik mező nem üres-e?

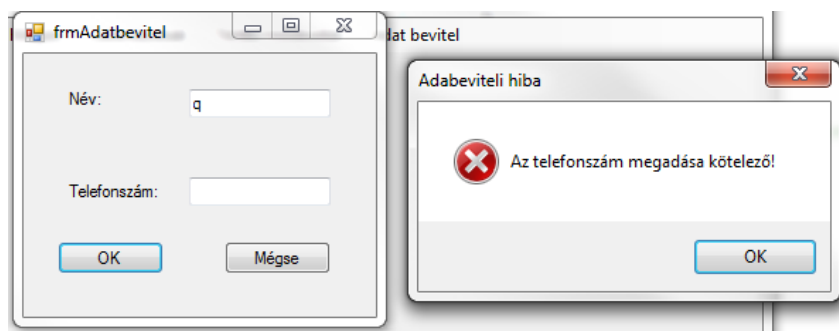
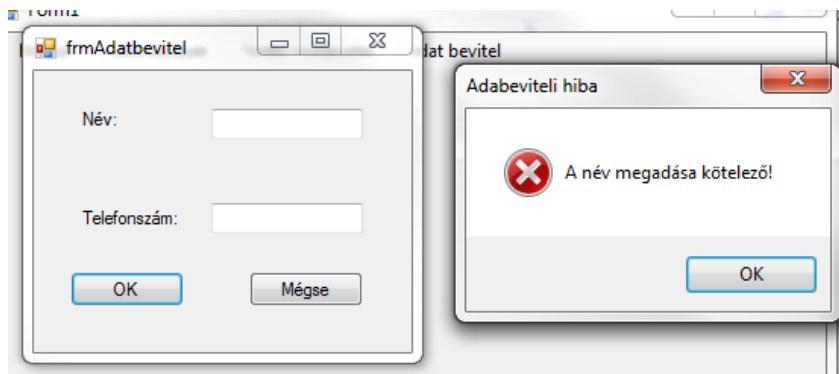
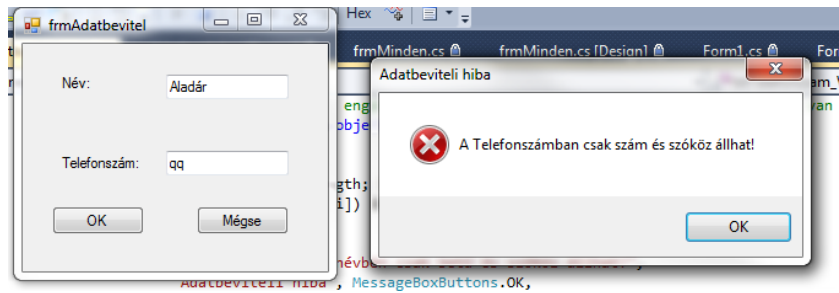
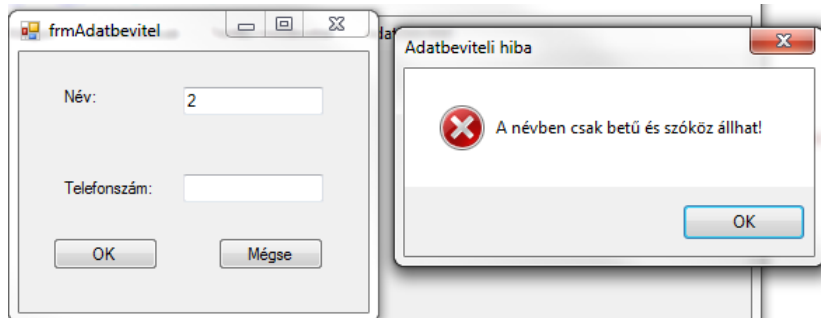
```
private void btnBevitel_Click(object sender, EventArgs e)
{
    bool VanHiba = false;
    if (txtNev .Text .Length == 0)
    {
        MessageBox.Show("A név megadása kötelező!", "Adabeviteli",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        VanHiba = true;
    }
    if (txtTelefonszam .Text .Length == 0)
    {
        MessageBox.Show("Az telefonszám megadása kötelező!",
            "Adabeviteli hiba",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        VanHiba = true;
    }

    if (!VanHiba)
    {
        DialogResult = DialogResult.OK;
        this.Close();
    }
}
```

18. A fő formon (Form1) az „Adatbevitel” menüponthoz rendeljük az frmAdatbevitel form megnyitását:

```
private void adatBevitelToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmAdatbevitel form = new frmAdatbevitel();
    form.Show();
}
```

19. Ellenőrizzük a beviteli form működését:



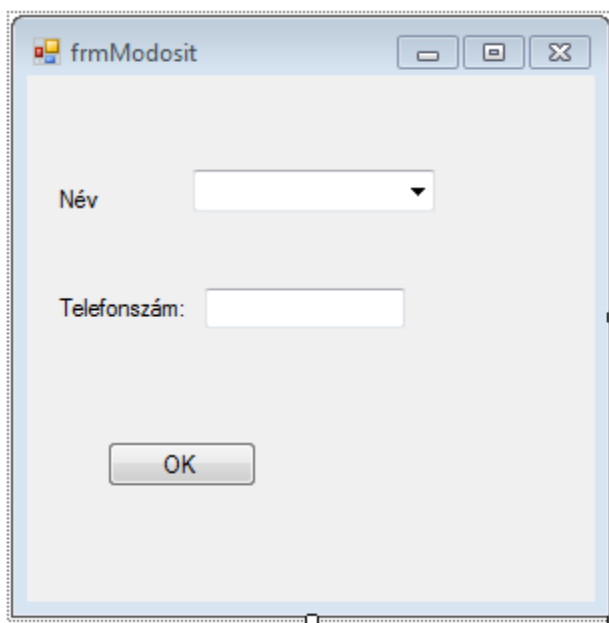
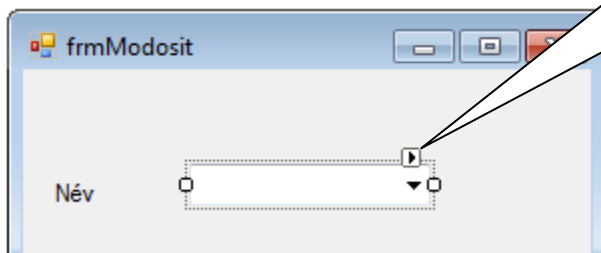
20. Kiegészítjük a Form1 kódjában az adatbevitel menüpont eddigi kódját azért, hogy a beviteli form (frmBevitel) OK gombjának megnyomásának hatására végrehajtsa a mentést.

```
ate void adatBevitelToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmAdatBevitel form = new frmAdatBevitel();
    if (form.ShowDialog() == DialogResult.OK) //ha az !!!OK felíratú gombot megnyomjuk
    {
        try
        {
            OleDbConnection kapcsolat = new OleDbConnection();//adatbázis kapcsolat objektum
            kapcsolat.ConnectionString = connectionString;//kapcsolati string definiálása
            OleDbCommand beviteliParancs = new OleDbCommand("INSERT INTO partner(Név,Telefonszám)" + "VALUES ('" + form.Név + "','" + form.Telefonszám + "')", kapcsolat);
            beviteliParancs.Connection.Open();// Kapcsolat megnyitása

            int temp = beviteliParancs.ExecuteNonQuery(); // SQL parancs végrehajtása
            if (temp > 0)
            {
                MessageBox.Show("Rendben! Adatbevitel rendben!");
            }
            else
            {
                MessageBox.Show("Adatbeviteli hiba!");
            }
            kapcsolat.Close();// Kapcsolat lezárása
        }
        catch (Exception kivétel)
        {
            MessageBox.Show(kivétel.Message, "Adatbázis hiba!!!",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
form.Show();
```

21. Bevitel ellenőrzése.

22. Módosítás form létrehozása: A név mezőnél ComboBox-t használunk.

The image shows a Windows Form titled 'frmModosit'. It contains two input fields: 'Név' (Name) and 'Telefonszám' (Phone number). The 'Név' field is currently empty. Below the input fields is an 'OK' button. The form has a standard Windows XP-style title bar with minimize, maximize, and close buttons.This image shows the same 'frmModosit' form, but the 'Név' field is now a ComboBox. A dashed border around the field indicates it is selected. A small arrow icon is visible on the right side of the field, indicating its dropdown functionality.

ComboBox Tasks

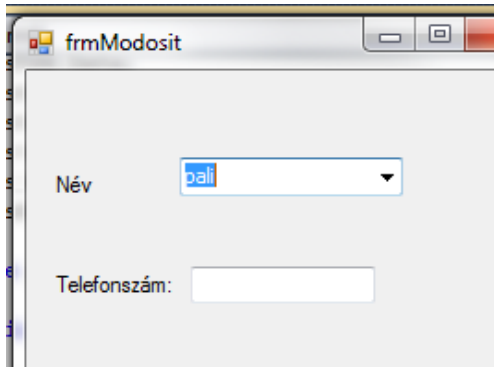
DataSource: partnerBindingSource

DisplayMember:Név

23. A fő formon (Form1) az „Módosítás” menüponthoz rendeljük az frmModosit form megnyitását:

```
private void módosításToolStripMenuItem_Click(c
{
    frmModosit form = new frmModosit();
    form.Show();
}
```

24. Ellenőrizzük a „ComboBox” működését:

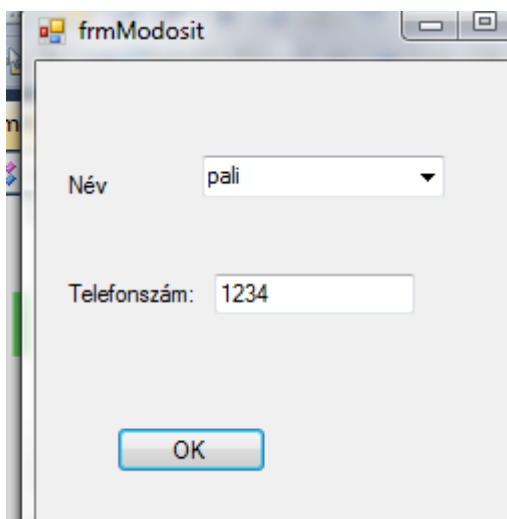


25. frmModosit formon hívjuk meg a „using System.Data.OleDb;” névteret majd illesszük be a globális „connectionstring” változót:

```
static string connectionstring = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=|DataDirectory|\partnerek.accdb";
```

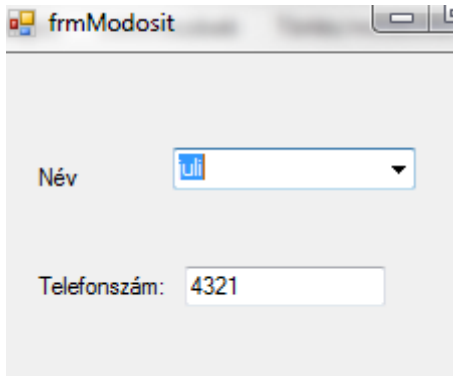
26. A frmModosit form Load eseményéhez rendeljük eseménykezelőt ami a ComboBox tartalma alapján kitölti a telefonszám mezőt is:

```
// TODO: This line of code loads data into the 'partnerekDataSet1.partner' table. You can move, or remove it, as needed.
this.partnerTableAdapter.Fill(this.partnerekDataSet1.partner);
OleDbConnection Kapcsolat = new OleDbConnection();//adatbázis kapcsolat objektum
Kapcsolat.ConnectionString = connectionstring;//kapcsolati string definiálása
Kapcsolat.Open();
OleDbCommand tszamKeres = new OleDbCommand("select telefonszám from partner where név='" + comboBox1.Text + "'", Kapcsolat);
txtTelefonszam.Text = tszamKeres.ExecuteScalar().ToString();
tszamKeres.Cancel();
Kapcsolat.Close();
```



27. Oldjuk meg, hogy a ComboBox tartalmát változtatva a telefonszám is változzon. Ehhez írjunk eseménykezelőt a comboBox1_SelectedIndexChanged eseményhez:

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    OleDbConnection Kapcsolat = new OleDbConnection();//adatbázis kapcsolat objektum
    Kapcsolat.ConnectionString = connectionstring;//kapcsolati string definiálása
    Kapcsolat.Open();
    OleDbCommand tszamKeres = new OleDbCommand("select telefonszám from partner where név='" + comboBox1.Text + "'", Kapcsolat);
    txtTelefonszam.Text = tszamKeres.ExecuteScalar().ToString();
    tszamKeres.Cancel();
    Kapcsolat.Close();
}
```



28. Írjunk tulajdonságokat a mezők kívülről történő elérésének biztosítására:

```
public string Nev
{
    get { return comboBox1.Text; }
}

//lekérdezhető a mező a formon kívülről is
public string Telefonszam
{
    get { return txtTelefonszam.Text; }
    set { txtTelefonszam.Text = value; }
}

//lekérdezhető, beállítható a mező a formon kívülről is
```

29. Készítsünk eseménykezelőt a a módosító form OK gombjának Click eseményéhez.

```
private void btnModosit_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.OK;
    this.Close();
}
```

30. Kiegészítjük a Form1 kódjában az Módosítás menüpont eddigi kódját azért, hogy a módosító form (frmModosit) OK gombjának megnyomásának hatására végrehajtsa a kiválasztott rekord módosítását:

```
private void módosításToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmModosit form = new frmModosit();
    if (form.ShowDialog() == DialogResult.OK)//ha az !!!OK felíratú gombot megnyomjuk
    {
        OleDbConnection Kapcsolat = new OleDbConnection();//adatbázis kapcsolat objektum
        Kapcsolat.ConnectionString = connectionString ;//kapcsolati string definiálása
        Kapcsolat.Open();
        OleDbCommand modositások = new OleDbCommand("update partner set Telefonszám = ('" + form.Telefonszam + "') where Név=('" + form.Név + "')", Kapcsolat)
        int temp = modositások.ExecuteNonQuery();
        if (temp > 0)
        {
            MessageBox.Show("Rendben! Módosítás mentve az adatbázisba.");
        }
        else
        {
            MessageBox.Show("Módosítási hiba!");
        }
        Kapcsolat.Close();
    }
}
```

31. Ellenőrizzük a módosító rendszer működését.

32. Rekord törlése: A törlést egy DataGridView rács segítségével oldjuk meg. A rácsban kiválasztott rekord azonosítója alapján végezzük a törlést az adatbázisból. Készítsün egy formot a törlés számára:

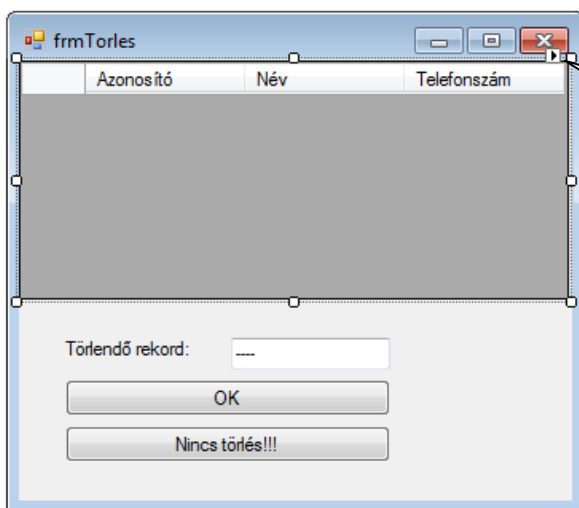
The screenshot shows a Windows form titled 'frmTorles'. It contains a table with three columns: 'Azonosító', 'Név', and 'Telefonszám'. The table body is currently empty. Below the table, there is a label 'Törendő rekord:' followed by a text input field containing '----'. At the bottom of the form, there are two buttons: 'OK' and 'Nincs törlés!!!'.

33. Készítsünk egy tulajdonságot az azonosító számára, hogy másik formból is lekérdezhető legyen.

```
public string Azonosito
{
    get { return txtAzonosito.Text; }
    //set { txtTelefonszam.Text = value; }
} //lekérdezhető, beállítható a mező a formon kívülről is
```

34. A frmTorles form „Load” eseményéhez rendeljük a rács feltöltését és itt állítsuk be a rácson belüli kijelölés módját is:

```
private void frmTorles_Load(object sender, EventArgs e)
{
    dataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
    // TODO: This line of code loads data into the 'partnerekDataSet3.partner' table. You
    this.partnerTableAdapter1.Fill(this.partnerekDataSet3.partner);
}
```

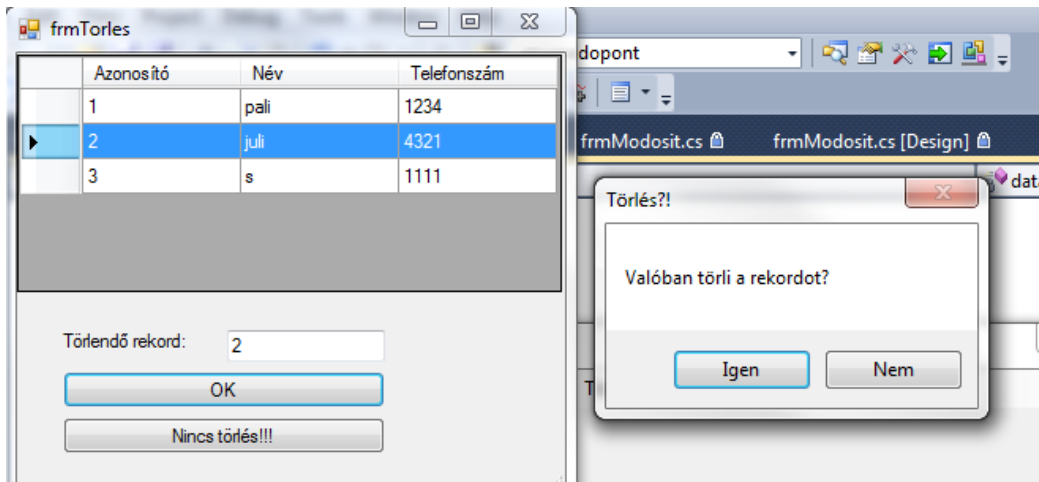


DataGridViewTasks

DataSource: partnerBindingSource...

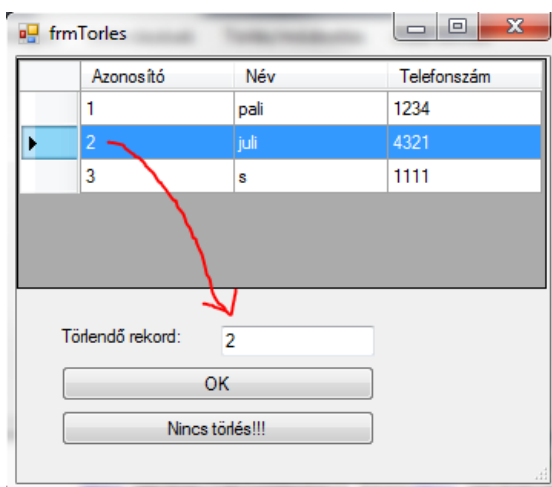
35. Az „OK” nyomógombhoz rendeljün biztonsági kérdést egy MessageBox-n keresztül:

```
private void btnTorol_Click(object sender, EventArgs e)
{
    DialogResult eredmeny= MessageBox.Show("Valóban törli a rekordot?", "Törlés?!", MessageBoxButtons.YesNo);
    if (eredmeny == DialogResult.Yes)
    { DialogResult = DialogResult.OK; }
    else { this.Close(); }
}
```



36. A rács CellClick eseményéhez kapcsoljuk a kiválasztott sor azonosítójának megszerzését.

```
private void dataGridView1_CellClick(object sender, DataGridViewCellEventArgs e)
{
    if (dataGridView1.Rows[e.RowIndex].Cells[0].Value != null)
    {
        txtAzonosito.Text = dataGridView1.Rows[e.RowIndex].Cells[0].Value.ToString();
    }
}
```



37. Fejlesztés (nem kötelező, de jobb): Elegendő egy helyen, globálisan létrehozni a `ConnectionString` változót, majd a program többi részén erre hivatkozni.

```
public class Connection
{
    public static string connectionstring = @"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=|DataDirectory|\partnerek.accdb";
}
```

hivatkozás a Form1-n: **Connection.connectionstring;**

hivatkozás a többi formon: **Form1. Connection.connectionstring;**

Ezekután több adatbázist is használhatunk és csak a `connectionstring`-t kell egyetlen helyen módosítani, vagy megoldható az adatbázis fájl programból történő választása is.

Tóth Tivadar