

Iskola szakmai vizsga gyakorló feladat

A `nevek.txt` állományban rögzítettük egy középiskola tanulóinak néhány adatát. Feltételezheti, hogy nincs két azonos nevű tanuló egy osztályban. Az állomány tartalma soronként:

- iskola kezdésének éve (2004-2007)
- az osztály betűjele (a-e)
- a diák neve (ékezetek nélkül).

Az adattagok pontosvesszővel vannak elválasztva. Példa (részlet) a `nevek.txt` állományra:

```
2004;d;Vavrek Kristof
2006;e;Hidas Reka
2006;d;Kun Michael
```

Megoldásában:

- A képernyőre írást igénylő részfeladatok eredményének megjelenítése előtt írja a képernyőre a feladat sorszámát (például: 3. feladat)!
- Az egyes feladatokban a kiírásokat a minta szerint készítse el!
- Az ékezetmentes kiírások is elfogadottak.
- Az azonosítókat kis- és nagybetűkkel is kezdheti.
- A program megírásakor az állományban lévő adatok helyes szerkezetét nem kell ellenőriznie, feltételezheti, hogy a rendelkezésre álló adatok a leírtaknak megfelelnek.
- A megoldását úgy készítse el, hogy az azonos szerkezetű, de tetszőleges bemeneti adatok mellett is helyes eredményt adjon! Oldja meg a következő feladatokat:

1. Készítsen `Iskola` néven konzol típusú projektet, melyben megoldja a következő feladatokat!

2. Olvassa be a `nevek.txt` állományban lévő adatokat! Hozzon létre megfelelő adatszerkezetet az adatok tárolására:

-Hozzon létre osztályt a tanulóknak. Az adattagok legyenek propertyk! (pld. `tanulo.nev...`)

Ha úgy érzi bővítse az osztályt olyan adattagokkal, amelyekkel a feladat megoldásához szükségesnek érzi! Az alapadatok csak konstruktoron keresztül legyenek módosíthatóak! (`public get`, `private set`)

-Az osztálynak legyen egy vagy több konstruktora, ami segít feldolgozni az adatokat! Pl megkapja a fájl 1 sorát, és beállítja a jellemzőket!

3. Írja ki a képernyőre, a tanulók adatait, illetve, hogy hány tanuló jár az iskolába!

4. Az iskolai rendszergazdának egyedi azonosítókat kell készítenie a számítógép-hálózat használatához. Az azonosítókat a következő módon alakítja ki: első karaktere az évfolyam utolsó számjegye (pl.: 2006 esetén 6), következő karakter az osztály betűjele, majd a vezetékneve első három karaktere, végül első keresztnéve első három karaktere következik. Az azonosítóban mindenütt kisbetűk szerepelnek. Feltételezhetjük, hogy a vezetéknev és az első keresztnév legalább 3 karakteres. Készítsen jellemzőt vagy függvényt, melyben

meghatározza a rendelkezésre álló adatokból a tanuló azonosítóját!

Az elkészített jellemzőt/függvényt felhasználva írja ki az adatszerkezetben tárolt első és utolsó tanuló azonosítóját a minta szerint!

5. Készítsen tesztet a függvényhez!

A teszt vizsgáljon hibás és nem hibás eredményeket is!

(A teszt a jelenlegi adatok vizsgálatára szól, független a fájl tartalmától! Tehát nem kell vizsgálnia, ha megváltozik valakinek az osztálya stb...)

Jó minta: Bodnar Szilvia →6cbodsz

Krizsan Vivien Evelin →6ckriviv

Rossz minta, bármi ami biztosan nem jó, tehát részben vagy egészében hibás átalakítás!

6. Írja ki az azonosító txt-be az összes tanuló nevét és azonosítóját!

7. Alkalmazza a tiszta kód elveit! Kommentelje a programot!

WPF-es feladat

Készítsen grafikus wpf-es alkalmazást a következő feladatok megoldására, melynek projektjét `IskolaWPF` néven mentse el!

9. Az alkalmazás grafikus felhasználói felületét alakítsa ki a minta szerint! Az ablak címsorában a „IskolaWPF” szöveget jelenítse meg!

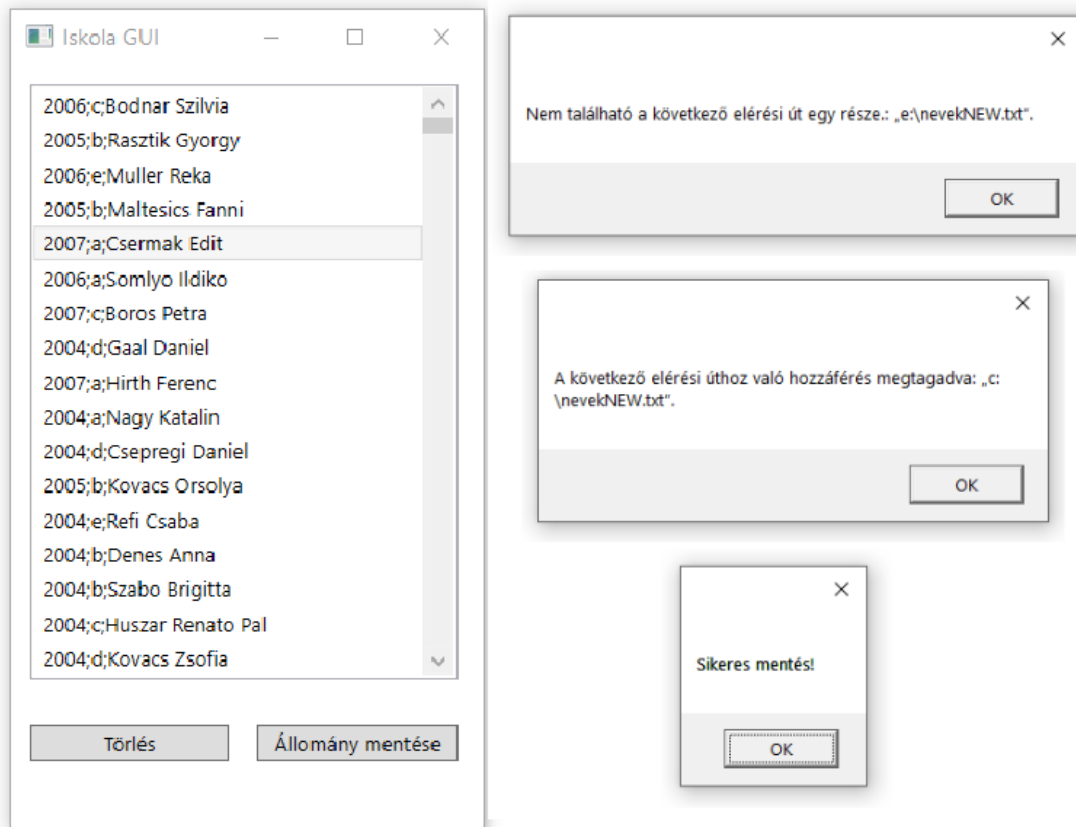
10. Az ablakon található listába a program induláskor töltse be a `nevek.txt` állomány sorait! A lista elemei a forrásfájl egy-egy sora (továbbiakban tanulója) legyen!

Az adatok tárolásához nyugodtan használja az előző feladatban megírt osztályokat, és tárolja hasonlóan listában, vagy más alkalmas adatszerkezetben az adatokat!

11. Oldja meg, hogy a kijelölt tanuló a „Törlés” parancsgomb lenyomása után törlésre kerüljön a listából! Ha a listában nincs kijelölt tanuló, akkor törléskor a „Nem jelölt ki tanulót!” szöveg jelenjen meg egy felugró ablakban!

12. Ha az „Állomány mentése” parancsgombra kattintunk, akkor történjen meg a listából a tanulók mentése a `nevekNEW.txt` állományba, melynek szerkezete a forrásállomány szerinti legyen! Ha a mentés sikeres volt, akkor a „Sikeres mentés!” felirat jelenjen meg egy felugró ablakban! Ha az állomány mentése sikertelen, akkor a hibaüzenet (a hibához tartozó beépített üzenet/message) jelenjen meg egy felugró ablakban! Lehetséges hibaokokhoz tartozó beépített üzeneteket a minták között talál!

Minták a grafikus alkalmazás futására:



A feladat videólinkje:

<https://www.youtube.com/watch?v=YKOry51oiz0&t=98s> Konzolos (CLI) rész

<https://www.youtube.com/watch?v=wcBQEb3Xd2w> Grafikus (GUI) rész

<https://www.youtube.com/watch?v=0MZULSGqXXo&list=PLqpCLCk7TsMO6JEqDN67hV1s6EWjdi9&index=6> Konzolos (CLI) és WPF-es rész. A WPF-es rész a videó 29. percétől kezdődik!!!!

A feladat konzolos (CLI) kódja órai munka alapján

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; // Legelső dolog, hogy ezt a névteret felvesszük, mert ez kell az adatokkal való munkához!!!

namespace Iskola
{
    public class Program
    {
        static void Main(string[] args)
        {
            List<tanulo> tanulolista = new List<tanulo>(); // Ennél a pontnál létre kell hoznunk egy tanulo osztályt külön tanulo.cs file-ban a Solution Explorerben, mert enélkül piros szaggatott vonallal alá fogja húzni, hibát jelez. A tanulo.cs file-nak tartalmaznia kell azt, amit a nevek.txt file tartalmaz, mert ez lesz a beolvasandó file, ami a tanulók adatait tartalmazza. Pontosabban a tanulo kezdési évét az iskolában, az osztálya betűjelét, illetve a nevét. A tanulok.cs file létrehozása: Solution Explorerben jobb klikk Iskola --> Add --> Class. A Class-t, azaz az osztályt elnevezzük tanulo.cs-nek és OK. Innentől kezdve a List után nem jelez hibát.

            StreamReader sr = new StreamReader("nevek.txt"); // Beolvassuk a nevek.txt file tartalmát.

            while (!sr.EndOfStream) // Amíg a beolvasás nem ér a végére a nevek.txt file-nak, míg be nem olvasta az összes adatot belőle, addig
            {
                tanulolista.Add(new tanulo(sr.ReadLine())); // adja hozzá a tanulolista nevű listához a nevek.txt file-ban található összes tanulót és írja ki az sr nevű beolvasás eredményét külön sorokban és jelenítse is meg azokat!
            }
            sr.Close(); // Ha a beolvasás a végére ért zárja be az sr nevű beolvasást.

            // Tanulók minden adata és darabszáma

            Console.WriteLine("3. feladat: A tanulók száma és minden adatuk.");
            Console.WriteLine();

            foreach (var item in tanulolista)
            {
                Console.WriteLine("A tanuló neve: " + item.DiakNeve);
                Console.WriteLine("A tanuló kezdési éve: " + item.KezdesEve);
                Console.WriteLine("A tanuló osztálya: " + item.OsztalyBetujele);
                Console.WriteLine();
            }
            Console.WriteLine("A tanulók száma: " + tanulolista.Count + " fő.");
        }
    }
}
```

// 4. feladat: a névsorban az első és az utolsó tanuló azonosítójának létrehozása. Az azonosítók felépítése: első karaktere a kezdés évének utolsó számjegye, második karakter a tanuló vezetéknévének első három betűje, harmadik karaktere keresztnévének első három karaktere.

```
        Console.WriteLine();
        Console.WriteLine("4. feladat: a névsorban első és utolsó tanuló
egyedi azonosítója:");
        Console.WriteLine();

        Console.WriteLine(tanulolista[0].DiakNeve); // Ezzel a sorral
határozzuk meg a névsorban az első tanuló nevét.
        Console.WriteLine(createazonosito(tanulolista[0]));

        Console.WriteLine(tanulolista[tanulolista.Count-1].DiakNeve); //
Ezzel a sorral kapjuk meg a névsorban az utolsó tanuló nevét.
        Console.WriteLine(createazonosito(tanulolista[tanulolista.Count-1]));

        // Írja ki az összes tanuló nevét és azonosítóját a nevek2.txt fájlba.

        StreamWriter sw = new StreamWriter("nevek2.txt");

        foreach (tanulo t in tanulolista)
        {
            string sor = t.DiakNeve + "#" + createazonosito(t);
            sw.WriteLine(sor);
        }
        sw.Close();

        Console.Read();
    }

    public static string createazonosito(tanulo t)
    {
        string azonosito = "";
        azonosito += (t.KezdesEve % 10).ToString();
        azonosito += (t.OsztalyBetujele);
        azonosito += (t.DiakNeve.Substring(0, 3).ToLower());
        azonosito += t.DiakNeve.Split(' ')[1].Substring(0, 3).ToLower();

        return azonosito;
    }
}
```

tanulo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iskola
{
    public class tanulo
    {
        public int KezdesEve { get; private set; } // A nevek.txt beolvasandó
        file első oszlopa a tanuló iskolában való kezdésének évszámát tartalmazza. Egy
        szám, ami ezért integer (int) típusú lesz. A private set azt jelenti, hogy az
        adat módosításának lehetőségét elrejtjük a felhasználó elől. Ezt Encapsulation-
        nek (kapszulázás) nevezzük. Ezeket a privát változókat csak ugyanazon az osztályon
        belül lehet elérni (egy külső osztály nem fér hozzá). A { get; private set } a
        függvény konstruktora.

        public string OsztalyBetujele { get; private set; }
        public string DiakNeve { get; private set; }

        public tanulo (string sor) // A tanulók adatait nyilvánossá (public) kell
        tenni, hogy a felhasználó is láthassa őket. A string sor-ral beolvasunk egy sort
        a nevek.txt file-ból. A sor nevű szöveges változóban tárolunk egy sornyi adatot a
        nevek.txt file-ból, ami egy tanuló adatait (iskolában való kezdésének évszámát,
        osztályának betűjelét és teljes nevét) tartalmazza.
        {
            string[] darabok = sor.Split(';'); // Nekünk nem csak egy, hanem sok
            tanulo adatait kell tárolnunk, ezért létre kell hozni egy egyelőre ismeretlen
            nagyságú, több elemű szöveges tömböt (string[]) és elnevezzük darabok-nak. Ebben
            a nevek.txt file-ban szereplő összes tanuló adatai majd benne lesznek. Viszont a
            sor nevű változóban itt már a nevek.txt file összes sora benne lesz, amiket a
            kettőspontok mentén el kell darabolni (Splittelni).

            this.KezdesEve = Convert.ToInt32(darabok[0]); // A this kulcsszó az
            aktuális tanulo osztály egy példányára utal. Itt a KezdesEve int, azaz egy egész
            számos változóként lett megadva, de hogy a fordító ezt értelmezni tudja, át kell
            konvertálni szöveges (stringes) változóra. Ez lesz a tömb 0. eleme.

            this.OsztalyBetujele = darabok[1];
            this.DiakNeve = darabok[2];
        }
    }
}
```

A tanulo.cs file-ban a kurzort a public class tanulo-ra visszük, majd jobb klikk. Creat Unit Test menüpont kiválasztása, klikk, a felugró ablakot leokézzuk csak. Megcsinálja automatikusan a Unit Test-et. Az Assert.Fail helyére beírjuk a mi kis ötsoros kódunkat. Aztán Test menüpont legördülő listájában Run All Test. Ha minden okés, akkor zöld színű pipák fognak megjelenni a tesztablakban. Fontos, hogy csak public class-ekre enged tesztelni, private-ra nem!!!!

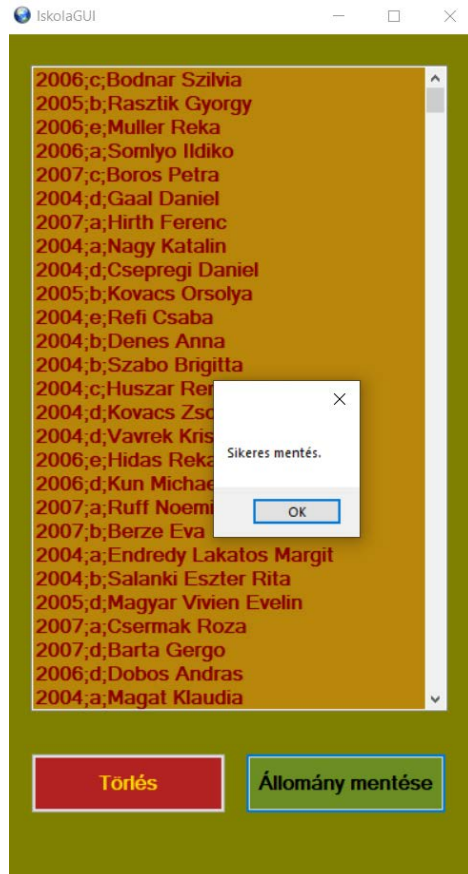
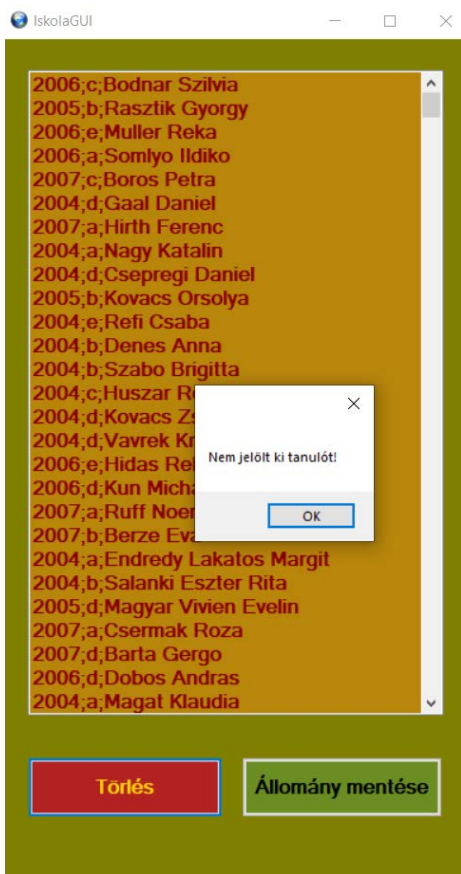
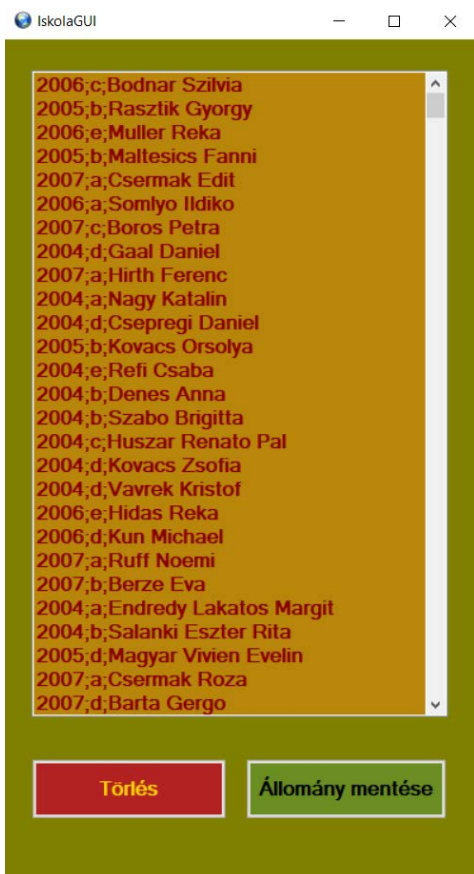
ProgramTests.cs

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Iskola;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iskola.Tests
{
    [TestClass()]
    public class ProgramTests
    {
        [TestMethod()]
        public void createazonositoTest()
        {
            tanulo t = new tanulo(@"2006;c;Bodnar Szilvia");
            Assert.AreEqual("6cbodszi", Program.createazonosito(t));
            tanulo t2 = new tanulo(@"2006;c;Krizsan Vivien Evelin");
            Assert.AreEqual("6ckriviv", Program.createazonosito(t2));
            Assert.AreNotEqual("8cbodszi", Program.createazonosito(t));
        }
    }
}
```

A feladat grafikus felületű, Form-os (GUI) kódja órai munka alapján

A program futás közben:



Form1.cs

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

namespace IskolaGUI
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            StreamReader sr = new StreamReader("nevekGUI.txt", Encoding.UTF8);
            string sor = "";
            while(!sr.EndOfStream)
            {
                sor = sr.ReadLine();
                listBox1.Items.Add(sor);
            }

            sr.Close();
        }

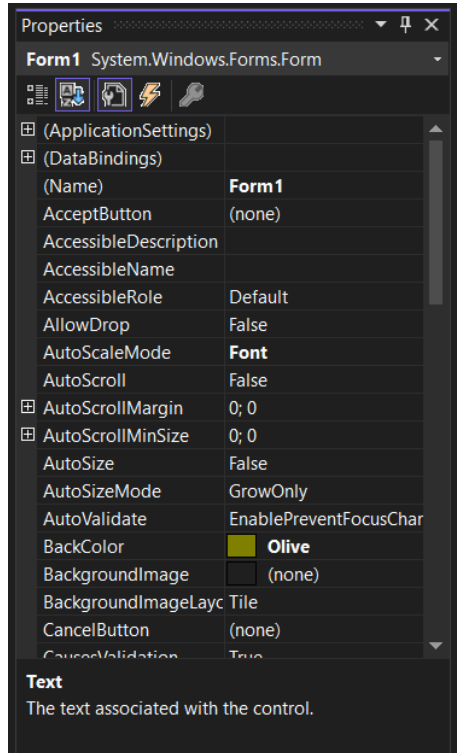
        private void button1_Click(object sender, EventArgs e)
        {
            if(listBox1.SelectedIndex== -1) // A -1 egy azt jelzi, hogy nincs a listBox1-ben kijelölve
            {
                MessageBox.Show("Nem jelölt ki tanulót!");
            }

            else
            {
                int hanyadik = listBox1.SelectedIndex;
                listBox1.Items.RemoveAt(hanyadik);
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                StreamWriter sw = new StreamWriter("nevekNEW.txt"); //( "c:\\valami\\nevekNEW.txt");
                foreach(var elem in listBox1.Items)
                {
                    sw.WriteLine(elem);
                }

                sw.Close();
                MessageBox.Show("Sikeres mentés.");
            }
            catch(Exception hiba)
            {
                MessageBox.Show(hiba.Message);
            }
        }
    }
}
```

A Form1 tulajdonságai:



Properties

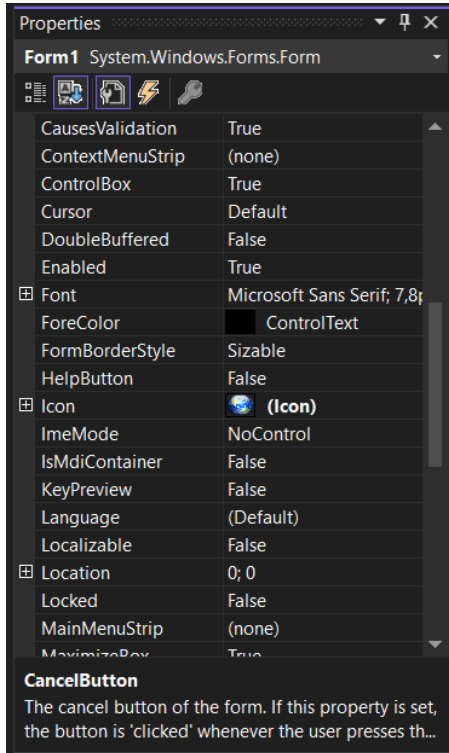
Form1 System.Windows.Forms.Form

(DataBindings)

(Name)	Form1
AcceptButton	(none)
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
AutoScaleMode	Font
AutoScroll	False
AutoScrollMargin	0; 0
AutoScrollMinSize	0; 0
AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusChar
BackColor	Olive
BackgroundImage	(none)
BackgroundImageLayout	Tile
CancelButton	(none)
CausesValidation	True

Text

The text associated with the control.



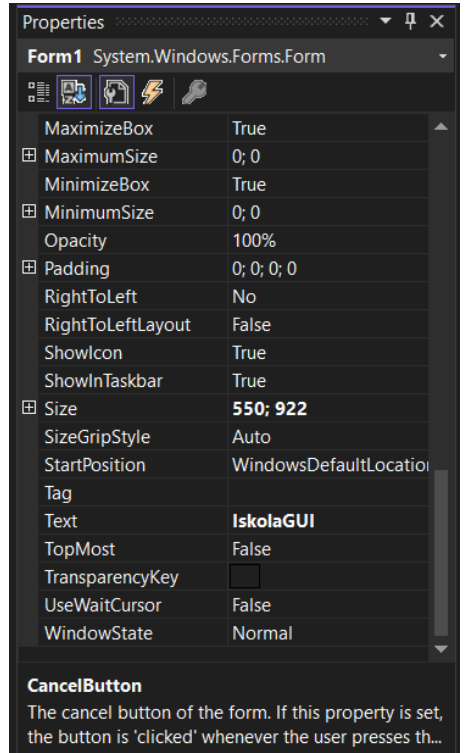
Properties

Form1 System.Windows.Forms.Form

CausesValidation	True
ContextMenuStrip	(none)
ControlBox	True
Cursor	Default
DoubleBuffered	False
Enabled	True
Font	Microsoft Sans Serif; 7,8
ForeColor	ControlText
FormBorderStyle	Sizable
HelpButton	False
Icon	(Icon)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False
Language	(Default)
Localizable	False
Location	0; 0
Locked	False
MainMenuStrip	(none)
MaximizeBox	True

CancelButton

The cancel button of the form. If this property is set, the button is 'clicked' whenever the user presses th...



Properties

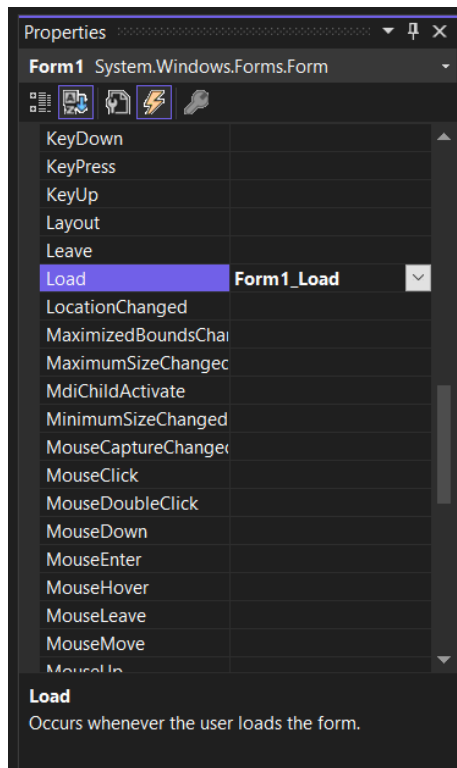
Form1 System.Windows.Forms.Form

MaximizeBox	True
MaximumSize	0; 0
MinimizeBox	True
MinimumSize	0; 0
Opacity	100%
Padding	0; 0; 0; 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
Size	550; 922
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	IskolaGUI
TopMost	False
TransparencyKey	
UseWaitCursor	False
WindowState	Normal

CancelButton

The cancel button of the form. If this property is set, the button is 'clicked' whenever the user presses th...

A Form1 eseménykezelője:



Properties

Form1 System.Windows.Forms.Form

KeyDown	
KeyPress	
KeyUp	
Layout	
Leave	
Load	Form1_Load
LocationChanged	
MaximizedBoundsChanged	
MaximumSizeChanged	
MdiChildActivate	
MinimumSizeChanged	
MouseCaptureChanged	
MouseClick	
MouseDoubleClick	
MouseDown	
MouseEnter	
MouseHover	
MouseLeave	
MouseMove	
MouseUp	

Load

Occurs whenever the user loads the form.

A listBox1 tulajdonságai:

Properties

listBox1 System.Windows.Forms.ListBox

(ApplicationSettings)

(DataBindings)

(Name) listBox1

AccessibleDescription

AccessibleName

AccessibleRole Default

AllowDrop False

Anchor Top, Left

BackColor DarkGoldenrod

BorderStyle Fixed3D

CausesValidation True

ColumnWidth 0

ContextMenuStrip (none)

Cursor Default

DataSource (none)

DisplayMember (none)

Dock None

DrawMode Normal

Edit Items...

Items

The items in the list box.

Properties

listBox1 System.Windows.Forms.ListBox

DrawMode Normal

Enabled True

Font Microsoft Sans Serif; 12

ForeColor DarkRed

FormatString

FormattingEnabled True

GenerateMember True

HorizontalExtent 0

HorizontalScrollbar False

ImeMode NoControl

IntegralHeight True

ItemHeight 25

Items (Collection)

Location 31; 32

Locked False

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Edit Items...

DrawMode

Controls list box painting. Either the system [NORMAL] or the user [OWNERDRAW] paints each item.

Properties

listBox1 System.Windows.Forms.ListBox

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers Private

MultiColumn False

RightToLeft No

ScrollAlwaysVisible False

SelectionMode One

Size 465; 679

Sorted False

TabIndex 0

TabStop True

Tag

UseTabStops True

UseWaitCursor False

ValueMember

Visible True

Edit Items...

MinimumSize

Specifies the minimum size of the control.

A Törlés (button1) gomb tulajdonságai:

Properties

button1 System.Windows.Forms.Button

(ApplicationSettings)

(DataBindings)

(Name) button1

AccessibleDescription

AccessibleName

AccessibleRole Default

AllowDrop False

Anchor Top, Left

AutoEllipsis False

AutoSize False

AutoSizeMode GrowOnly

BackColor Firebrick

BackgroundImage (none)

BackgroundImageLayout Tile

CausesValidation True

ContextMenuStrip (none)

Cursor Default

DialogResult None

Dock None

Enabled True

Text

The text associated with the control.

Properties

button1 System.Windows.Forms.Button

Enabled True

FlatAppearance

FlatStyle Standard

Font Microsoft Sans Serif; 12

ForeColor Gold

GenerateMember True

Image (none)

ImageAlign MiddleCenter

ImageIndex (none)

ImageKey (none)

ImageList (none)

Location 31; 745

Locked False

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers Private

Padding 0; 0; 0; 0

RightToLeft No

Size 218; 63

Enabled

Indicates whether the control is enabled.

Properties

button1 System.Windows.Forms.Button

Locked False

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers Private

Padding 0; 0; 0; 0

RightToLeft No

Size 218; 63

TabIndex 1

TabStop True

Tag

Text Törlés

TextAlign MiddleCenter

TextImageRelation Overlay

UseCompatibleTextRendering False

UseMnemonic True

UseVisualStyleBackColor False

UseWaitCursor False

Visible True

Size

The size of the control in pixels.

A Törlés (button1) gomb eseménykezelője:

Properties

button1 System.Windows.Forms.Button

(DataBindings)

AutoSizeChanged

BackColorChanged

BackgroundImageChar

BackgroundImageLayc

BindingContextChange

CausesValidationChang

ChangeUICues

Click

ClientSizeChanged

ContextMenuStripChar

ControlAdded

ControlRemoved

CursorChanged

DockChanged

DpiChangedAfterParer

DpiChangedBeforePar

DragDrop

DragEnter

DragLeave

button1_Click

Click

Occurs when the component is clicked.

Az Állomány mentése (button2) gomb tulajdonságai:

Properties

button2 System.Windows.Forms.Button

(ApplicationSettings)

(DataBindings)

(Name) button2

AccessibleDescription

AccessibleName

AccessibleRole Default

AllowDrop False

Anchor Top, Left

AutoEllipsis False

AutoSize False

AutoSizeMode GrowOnly

BackColor OliveDrab

BackgroundImage (none)

BackgroundImageLayc Tile

CausesValidation True

ContextMenuStrip (none)

Cursor Default

DialogResult None

Dock None

Enabled True

Text

The text associated with the control.

Properties

button2 System.Windows.Forms.Button

Enabled True

FlatAppearance

FlatStyle Standard

Font Microsoft Sans Serif; 1

ForeColor ControlText

GenerateMember True

Image (none)

ImageAlign MiddleCenter

ImageIndex (none)

ImageKey (none)

ImageList (none)

Location 271; 745

Locked False

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers Private

Padding 0; 0; 0; 0

RightToLeft No

Size 225; 63

Enabled

Indicates whether the control is enabled.

Properties

button2 System.Windows.Forms.Button

Lockedd False

Margin 3; 3; 3; 3

MaximumSize 0; 0

MinimumSize 0; 0

Modifiers Private

Padding 0; 0; 0; 0

RightToLeft No

Size 225; 63

TabIndex 2

TabStop True

Tag

Text Állomány mentése

TextAlign MiddleCenter

TextImageRelation Overlay

UseCompatibleTextRei False

UseMnemonic True

UseVisualStyleBackCol False

UseWaitCursor False

Visible True

Size

The size of the control in pixels.

Az Állomány mentése (button2) gomb eseménykezelője:

Properties

button2 System.Windows.Forms.Button

(DataBindings)

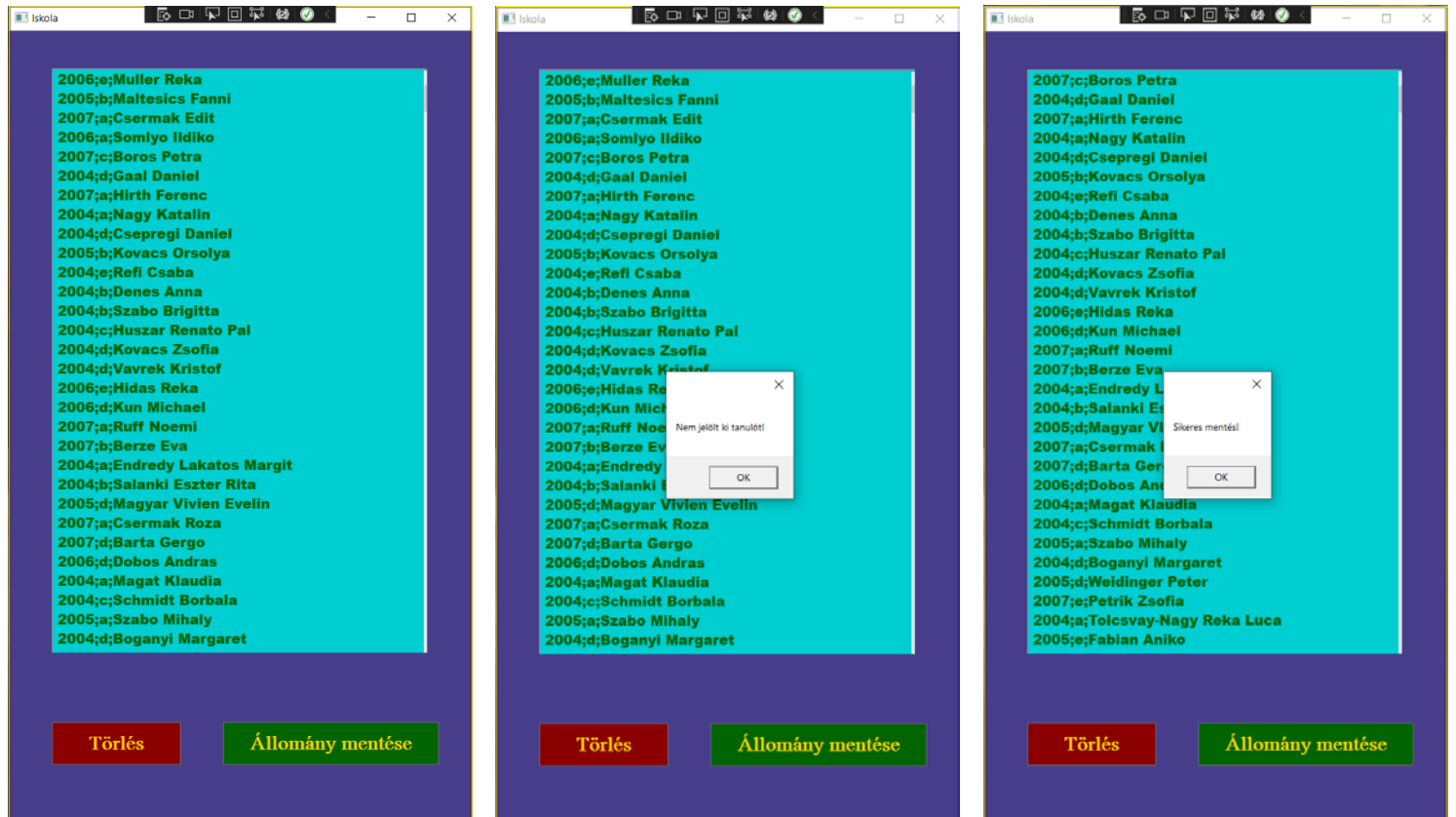
AutoSizeChanged	
BackColorChanged	
BackgroundImageChar	
BackgroundImageLayc	
BindingContextChange	
CausesValidationChang	
ChangeUICues	
Click	button2_Click
ClientSizeChanged	
ContextMenuStripChar	
ControlAdded	
ControlRemoved	
CursorChanged	
DockChanged	
DpiChangedAfterParer	
DpiChangedBeforePar	
DragDrop	
DragEnter	
DragLeave	

Click

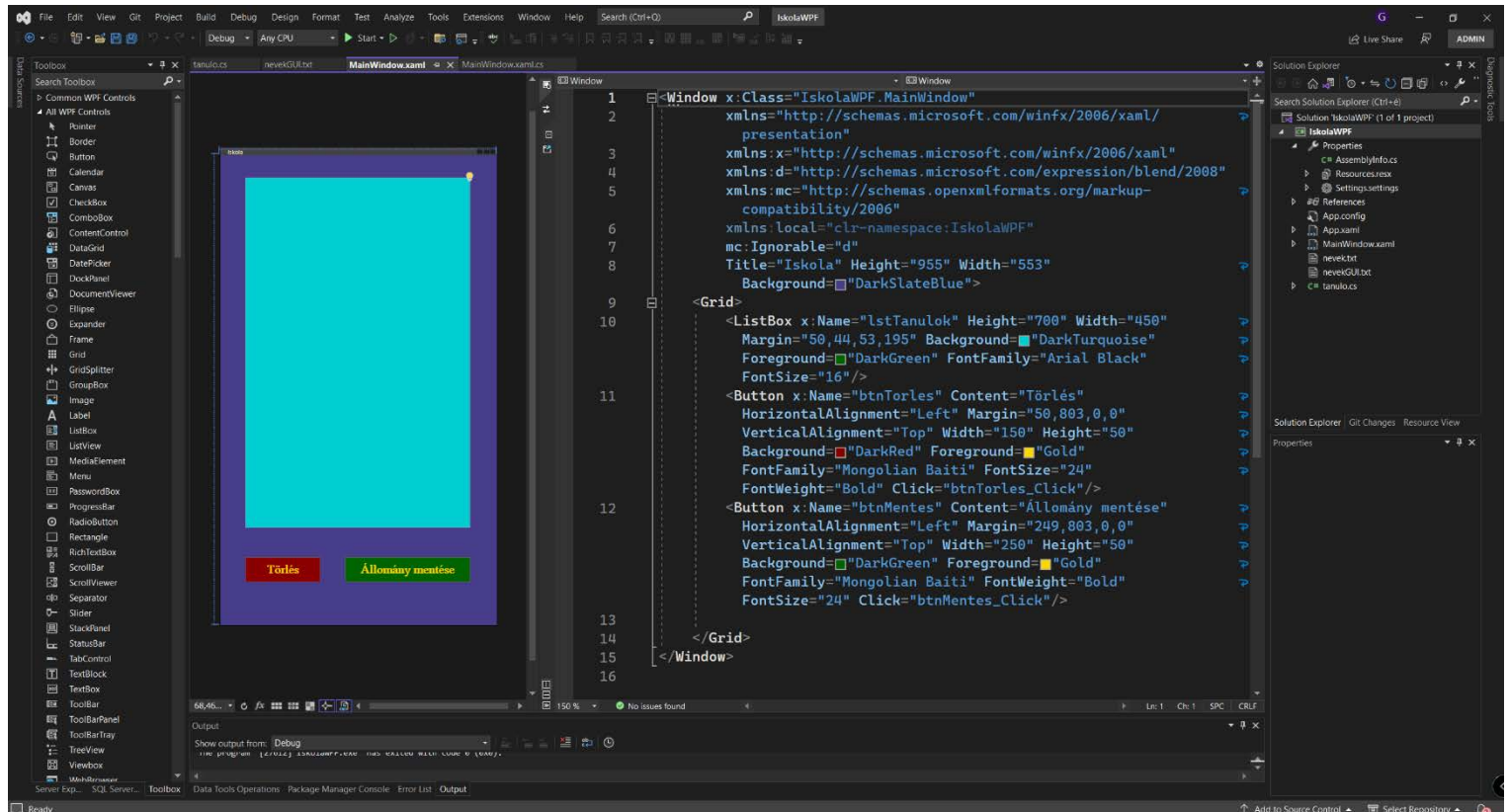
Occurs when the component is clicked.

A feladat WPF kódja órai munka alapján

IskolaWPF futás közben:



Az Iskola WPF tulajdonságai (MainWindow.xaml):



Az IskolaWPF tulajdonságai kóddal(MainWindow.xaml):

```
<Window x:Class="IskolaWPF.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:IskolaWPF"
        mc:Ignorable="d"
        Title="Iskola" Height="955" Width="553" Background="DarkSlateBlue">
    <Grid>
        <ListBox x:Name="lstTanulok" Height="700" Width="450" Margin="50,44,53,195"
        Background="DarkTurquoise" Foreground="DarkGreen" FontFamily="Arial Black" FontSize="16"/>
        <Button x:Name="btnTorles" Content="Törlés" HorizontalAlignment="Left" Margin="50,803,0,0"
        VerticalAlignment="Top" Width="150" Height="50" Background="DarkRed" Foreground="Gold"
        FontFamily="Mongolian Baiti" FontSize="24" FontWeight="Bold" Click="btnTorles_Click"/>
        <Button x:Name="btnMentes" Content="Állomány mentése" HorizontalAlignment="Left"
        Margin="249,803,0,0" VerticalAlignment="Top" Width="250" Height="50" Background="DarkGreen"
        Foreground="Gold" FontFamily="Mongolian Baiti" FontWeight="Bold" FontSize="24"
        Click="btnMentes_Click"/>

    </Grid>
</Window>
```

Az IskolaWPF teljes kódja (MainWindow.xaml.cs):

```
using Iskola;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace IskolaWPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();

            StreamReader sr = new StreamReader("nevekGUI.txt", Encoding.UTF8);

            while (!sr.EndOfStream)
            {
                lstTanulok.Items.Add(sr.ReadLine());
            }
            sr.Close();
        }

        private void btnTorles_Click(object sender, RoutedEventArgs e)
        {
            if (lstTanulok.SelectedItem != null)
            {
                lstTanulok.Items.Remove(lstTanulok.SelectedItem);
            }
            else
            {
                MessageBox.Show("Nem jelölt ki tanulót!");
            }
        }
    }
}
```



```

    }

    private void btnMentes_Click(object sender, RoutedEventArgs e)
    {
        try
        {
            StreamWriter sw = new StreamWriter("nevekNEW.txt");
            foreach (var item in lstTanulok.Items)
            {
                sw.WriteLine(item);
            }
            sw.Close();

            MessageBox.Show("Sikeres mentés!");
        }

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
}
}

```

tanulo.cs kódja:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Iskola
{
    public class tanulo
    {
        public int KezdesEve { get; private set; } // A nevek.txt beolvasandó file első oszlopa a
        tanuló iskolában való kezdésének évszámát tartalmazza. Egy szám, ami ezért integer (int) típusú lesz.
        A private set azt jelenti, hogy az adat módosításának lehetőségét elrejtjük a felhasználó elől. Ezt
        Encapsulation-nek (kapszulázás) nevezzük. Ezeket a privát változókat csak ugyanazon az osztályon belül
        lehet elérni (egy külső osztály nem fér hozzá). A { get; private set } a függvény konstruktora.

        public string OsztalyBetujele { get; private set; }
        public string DiakNeve { get; private set; }

        public tanulo(string sor) // A tanulok adatait nyilvánossá (public) kell tenni, hogy a
        felhasználó is láthassa őket. A string sor-ra beolvasunk egy sort a nevek.txt file-ból. A sor nevű
        szöveges változóban tárolunk egy sornyi adatot a nevek.txt file-ból, ami egy tanuló adatait
        (iskolában való kezdésének évszámát, osztályának betűjelét és teljes nevét) tartalmazza.
        {
            string[] darabok = sor.Split(';'); // Nekünk nem csak egy, hanem sok tanulo adatait kell
            tárolnunk, ezért létre kell hozni egy egyelőre ismeretlen nagyságú, több elemű szöveges tömböt
            (string[]) és elnevezzük darabok-nak. Ebben a nevek.txt file-ban szereplő összes tanuló adatai majd
            benne lesznek. Viszont a sor nevű változóban itt már a nevek.txt file összes sora benne lesz, amiket
            a kettőspontok mentén el kell darabolni (Splittelni).

            this.KezdesEve = Convert.ToInt32(darabok[0]); // A this kulcsszó az aktuális tanulo
            osztály egy példányára utal. Itt a KezdesEve int, azaz egy egész számos változóként lett megadva, de
            hogy a fordító ezt értelmezni tudja, át kell konvertálni szöveges (stringes) változóvá. Ez lesz a
            tömb 0. eleme.

            this.OsztalyBetujele = darabok[1];
            this.DiakNeve = darabok[2];
        }
    }
}

```


A feladat konzolos kódja Bende Atti Youtube videója alapján

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace bdc_IskolaKonzolos
{
    class JelszóGeneráló
    {
        private Random Rnd;

        public JelszóGeneráló(Random r)
        {
            Rnd = r;
        }

        public string Jelszó(int jelszóHossz)
        {
            string jelszó = "";
            while (jelszó.Length < jelszóHossz)
            {
                char c = (char)Rnd.Next(48, 123);
                if ((c >= '0' && c <= '9') || (c >= 'a' && c <= 'z'))
                {
                    jelszó += c;
                }
            }
            return jelszó;
        }
    }

    class Program
    {

```

```
static List<IskolaClass> iLista = new List<IskolaClass>();

static void Main(string[] args)
{
    //beolvasás
    StreamReader sr = new StreamReader("nevek.txt", Encoding.UTF8);
    string sor = ""; //nincs fejléc adatsor
    while (!sr.EndOfStream)
    {
        sor = sr.ReadLine();
        IskolaClass m = new IskolaClass(sor);
        iLista.Add(m);
    }

    Console.WriteLine("3. feladat");
    Console.WriteLine("A tanulók száma: " + iLista.Count);

    Console.WriteLine("4. feladat");
    int maxNevHossz = int.MinValue;

    for (int i = 0; i < iLista.Count; i++)
    {
        int szokozSzam = iLista[i].Nev.Split(' ').Length - 1;
        if (iLista[i].Nev.Length - szokozSzam > maxNevHossz)
        {
            maxNevHossz = iLista[i].Nev.Length - szokozSzam;
        }
    }

    Console.WriteLine("Maximális karakterek száma: " + maxNevHossz);
    for (int i = 0; i < iLista.Count; i++)
    {
        int szokozSzam = iLista[i].Nev.Split(' ').Length - 1;
        if (iLista[i].Nev.Length - szokozSzam == maxNevHossz)
        {
            Console.WriteLine(iLista[i].Nev);
        }
    }

    Console.WriteLine("5. feladat");
    Console.WriteLine("Első: " + iLista[0].Nev + ": " + iLista[0].azonosito());
    Console.WriteLine("Utolsó: " + iLista[iLista.Count-1].Nev + ": " + iLista[iLista.Count - 1].azonosito());
```

```

Console.WriteLine("6. feladat");

Console.WriteLine("Kérek egy azonosítót!");

string az = Console.ReadLine();

int j = 0;

bool megvan = false;

while (megvan == false && j < iLista.Count)
{
    if (iLista[j].azonosito() == az)
    {
        megvan = true;

        Console.WriteLine(iLista[j].Nev);
    }

    j++;
}

if (megvan == false)
{
    Console.WriteLine("Nincs ilyen tanuló!");
}

Console.WriteLine("7. feladat");

Random r = new Random();

int tanuloSorszam = r.Next(0, iLista.Count);

JelszóGeneráló jelszoGen = new JelszóGeneráló(r);

Console.WriteLine(iLista[tanuloSorszam].Nev);

Console.WriteLine(jelszoGen.Jelszó(8));

Console.ReadKey();
}
}
}

```

IskolaClass.cs fájl tartalma:

```

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

```

```
namespace bdc_IskolaKonzolos
{
class IskolaClass
{
//adattagok
private int ev;
private string osztaly;
private string nev;

//konstruktor
public IskolaClass(string sor)
{
string[] d = sor.Split(';');
ev = Convert.ToInt32(d[0]);
osztaly = d[1];
nev = d[2];
}

public int Ev { get => ev; set => ev = value; }
public string Osztaly { get => osztaly; set => osztaly = value; }
public string Nev { get => nev; set => nev = value; }

public string azonosito()
{
char evUtolso = ev.ToString()[3];
string vezNev = nev.Substring(0, 3);
string kerNev = nev.Split(' ')[1].Substring(0, 3);
return (evUtolso + osztaly + vezNev + kerNev).ToLower();
}
}
}
```

Jelszógeneráló

```
class JelszóGeneráló
{
private Random Rnd;

public JelszóGeneráló(Random r)
{
Rnd = r;
```

```
}

public string Jelszó(int jelszóHossz)
{
    string jelszó = "";
    while (jelszó.Length < jelszóHossz)
    {
        char c = (char)Rnd.Next(48, 123);
        if ((c >= '0' && c <= '9') || (c >= 'a' && c <= 'z'))
        {
            jelszó += c;
        }
    }
    return jelszó;
}
```