

Programozás C# nyelven

5. előadás

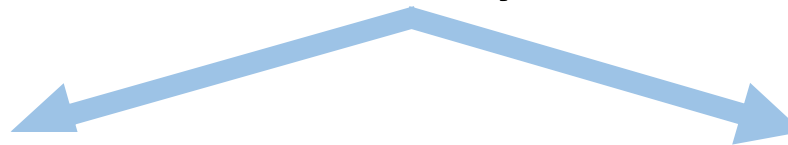


<http://e-learning.ujs.sk/>

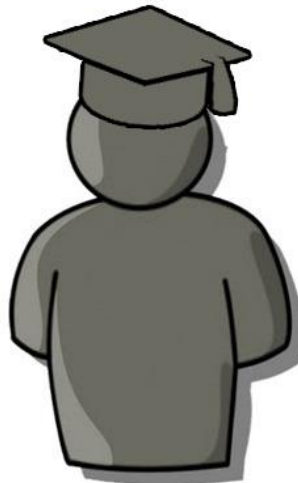
Nev
SzuletesiEv
Eletkor()



Szemely



Nev
SzuletesiEv
Eletkor()
_jegyek
_jegyekSzama
OsszesJegy()
UjJegy()
Átlag()



Diak

Nev
SzuletesiEv
Eletkor()
_fizetes
GetFizetes()
FizetesEmeles()
FizetesCsokkentes()



Munkas

A múlt heti feladat megoldása (Munkas.cs állomány):

014 OOP Feladat - Munkas

```
using System;
```

```
namespace _014_OOP_Feladat__Munkas
{
    public class Munkas : Szemely
    {
        private double _fizetes;

        public Munkas(string aNev, int aSzuletesiEv, double aFizetes) :
            base(aNev, aSzuletesiEv)
        {
            _fizetes = aFizetes;
        }

        public double GetFizetes()
        {
            return _fizetes;
        }
    }
}
```

```
public void FizetesEmeles(int szazalek)
{
    _fizetes = _fizetes + _fizetes * (szazalek / 100.0);
    _fizetes = Math.Round(_fizetes, 2);
}
```

```
public void FizetesCsokkentes(int szazalek)
{
    _fizetes = _fizetes - _fizetes * (szazalek / 100.0);
    _fizetes = Math.Round(_fizetes, 2);
}
```

```
}
```

```
}
```

```
private Szemely sz1, sz2;  
private Diak d1, d2;  
private Munkas m1, m2;
```

```
private void button1_Click(object sender, EventArgs e)
```

```
{  
    textBox1.Clear();  
    textBox1.AppendText("----- Objektumok létrehozása 1 ----- \n");  
    sz1 = new Szemely("Peti", 1996);  
    sz2 = new Szemely("Éva", 1999);  
    d1 = new Diak("Feri", 2003, new int[] { 1, 2, 2, 1, 2 });  
    d2 = new Diak("Timi", 2002, new int[] { 1, 1, 3, 1 });  
    m1 = new Munkas("Zoli", 1995, 590);  
    m2 = new Munkas("Tomi", 1997, 560);  
}
```

```
private void button2_Click(object sender, EventArgs e)
```

```
{  
    textBox1.AppendText("----- Életkorok kiírása 1 ----- \n");  
    textBox1.AppendText("Szemely: " + sz1.Nev + " most " + sz1.Eletkor() + " éves. \n");  
    textBox1.AppendText("Szemely: " + sz2.Nev + " most " + sz2.Eletkor() + " éves. \n");  
    textBox1.AppendText("Diak: " + d1.Nev + " most " + d1.Eletkor() + " éves. \n");  
    textBox1.AppendText("Diak: " + d2.Nev + " most " + d2.Eletkor() + " éves. \n");  
    textBox1.AppendText("Munkas: " + m1.Nev + " most " + m1.Eletkor() + " éves. \n");  
    textBox1.AppendText("Munkas: " + m2.Nev + " most " + m2.Eletkor() + " éves. \n");  
}
```

A tömbelemek típusát Szemely-nek deklaráltuk,
mert az őosztály mindig helyettesíthető
a belőle leszármaztatott osztállyal.

```
private Szemely[] tomb = new Szemely[6];
```

```
private void button3_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox1.AppendText("----- Objektumok létrehozása 2 -----\\n");
    tomb[0] = new Szemely("Peti", 1996);
    tomb[1] = new Szemely("Éva", 1999);
    tomb[2] = new Diak("Feri", 2003, new int[] { 1, 2, 2, 1, 2 });
    tomb[3] = new Diak("Timi", 2002, new int[] { 1, 1, 3, 1 });
    tomb[4] = new Munkas("Zoli", 1995, 590);
    tomb[5] = new Munkas("Tomi", 1997, 560);
}
```

```
private void button4_Click(object sender, EventArgs e)
{
    textBox1.AppendText("----- Életkorok kiírása 2 -----\\n");
    foreach (Szemely x in tomb)
    {
        textBox1.AppendText(x.GetType().Name + ": " + x.Nev + " most " +
            x.Eletkor() + " éves.\\n");
    }
}
```

Statikus és dinamikus típus:



Szemely x = new Szemely("Peti", 1996);

The diagram shows two red arrows originating from the underlined title. One arrow points to the variable 'x' in the code snippet, and the other points to the class name 'Szemely' in the same snippet.

x statikus típusa: Szemely

x dinamikus típusa: Szemely

Diak y = new Diak("Feri", 2003, new int[] {1, 2, 2, 1, 2});

y statikus típusa: Diak

y dinamikus típusa: Diak

Szemely z = new Diak("Timi", 2002, new int[] {1, 2, 3, 1});

z statikus típusa: Szemely

z dinamikus típusa: Diak

```
Szemely z = new Diak("Timi", 2002, new int[] {1, 2, 3, 1});
```

z statikus típusa: Szemely

z dinamikus típusa: Diak

Hogyan viselkedjen „z”? Mint egy „Szemely” vagy mint egy „Diak”?
Ha a Szemely-nek és a Diak-nak is lenne ugyanolyan nevű metódusa
(pl. „Vasarol()”), melyek különbözőképpen működnek, melyik fusson le?

Statikus és dinamikus kötés:

Statikus kötés: „fordítási idejű” vagy „korai” kötés (early binding),
fordítási időben dől el, hogy melyik metódus fut le.

Dinamikus kötés: „futási idejű” vagy „késői” kötés (late binding),
futási időben dől el, hogy melyik metódus fut le.


```
public class A
{
    public string StatikusKotes()
    {
        return "A osztály StatikusKotes() metódusa.";
    }

    public virtual string DinamikusKotes()
    {
        return "A osztály DinamikusKotes() metódusa.";
    }
}

public class B : A
{
    public new string StatikusKotes()
    {
        return "B osztály StatikusKotes() metódusa.";
    }

    public override string DinamikusKotes()
    {
        return "B osztály DinamikusKotes() metódusa.";
    }
}
```

A osztály

StatikusKotes()
DinamikusKotes()

**B osztály : A osztály**

StatikusKotes()
DinamikusKotes()

```
B x = new B();
x.StatikusKotes();
x.DinamikusKotes();
```

```
A x = new B();
x.StatikusKotes();
x.DinamikusKotes();
```



Virtual, override, new módosítók



B x = new B()

x.StatikusKotes()

x.DinamikusKotes()

A x = new B()

x.StatikusKotes()

(x as B).StatikusKotes()

x.DinamikusKotes()

B osztály StatikusKotes() metódusa.

B osztály DinamikusKotes() metódusa.

A osztály StatikusKotes() metódusa.

B osztály StatikusKotes() metódusa.

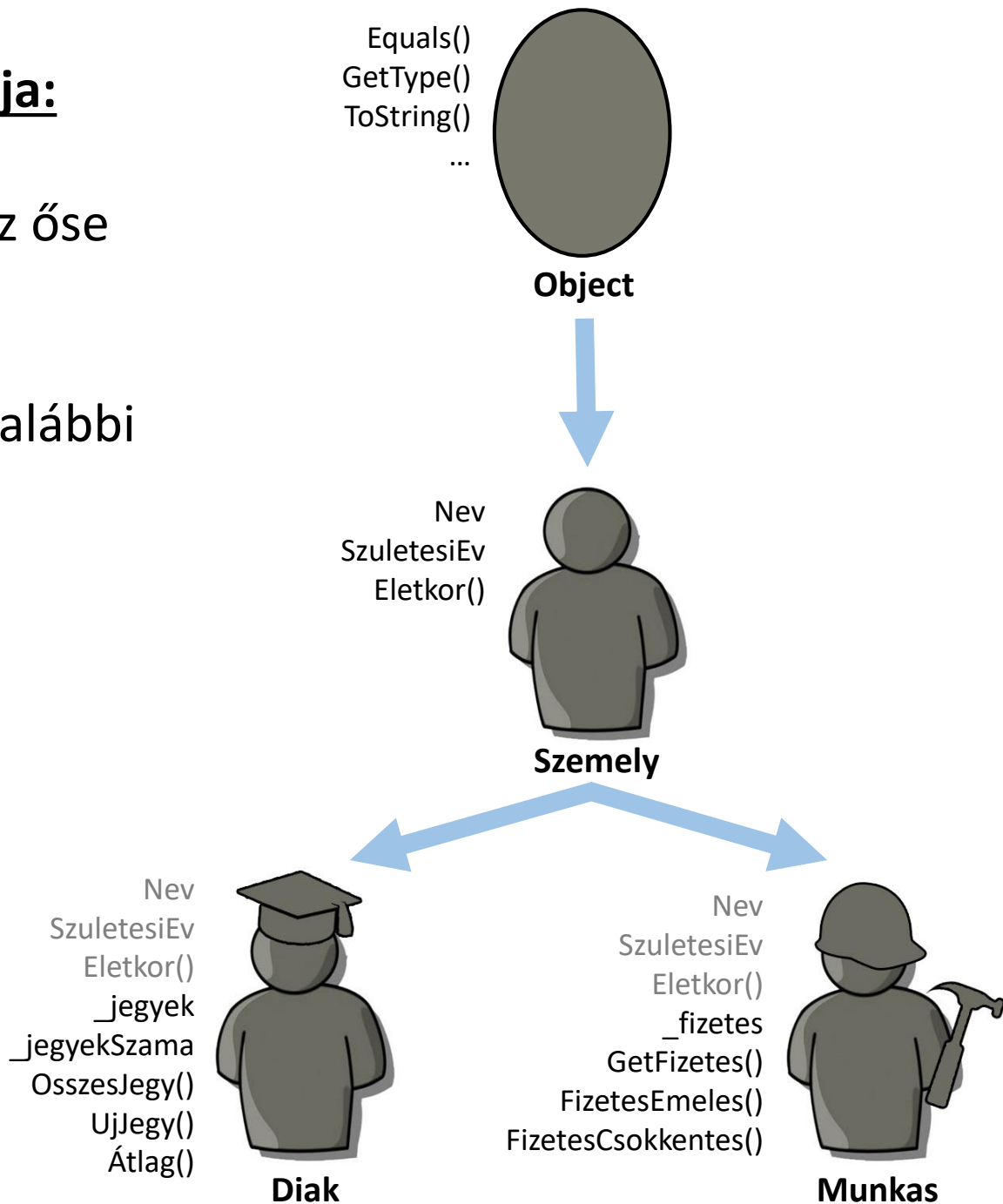
B osztály DinamikusKotes() metódusa.

Osztályok hierarchiája:

Minden osztálynak az őse az **Object** osztály.

Ez tartalmazza pl. az alábbi metódusokat:

- **Equals()**
- **GetType()**
- **ToString()**
- ...



ToString() metódot felülírása a Szemely osztályban:

017 OOP ToString metódot

```
public class Szemely
{
    public string Nev { get; }
    public int SzuletesiEv { get; }

    public Szemely(string aNev, int aSzuletesiEv)
    {
        Nev = aNev;
        SzuletesiEv = aSzuletesiEv;
    }

    public int Eletkor()
    {
        return DateTime.Now.Year - SzuletesiEv;
    }

    public override string ToString()
    {
        return Nev + " (" + SzuletesiEv + ")";
    }
}
```

```
private Szemely[] a = new Szemely[3];

private void Form1_Load(object sender, EventArgs e)
{
    a[0] = new Szemely("Peti", 1996);
    a[1] = new Diak("Feri", 2003, new int[] { 1, 2, 2, 1, 2 });
    a[2] = new Munkas("Zoli", 1995, 590);
}

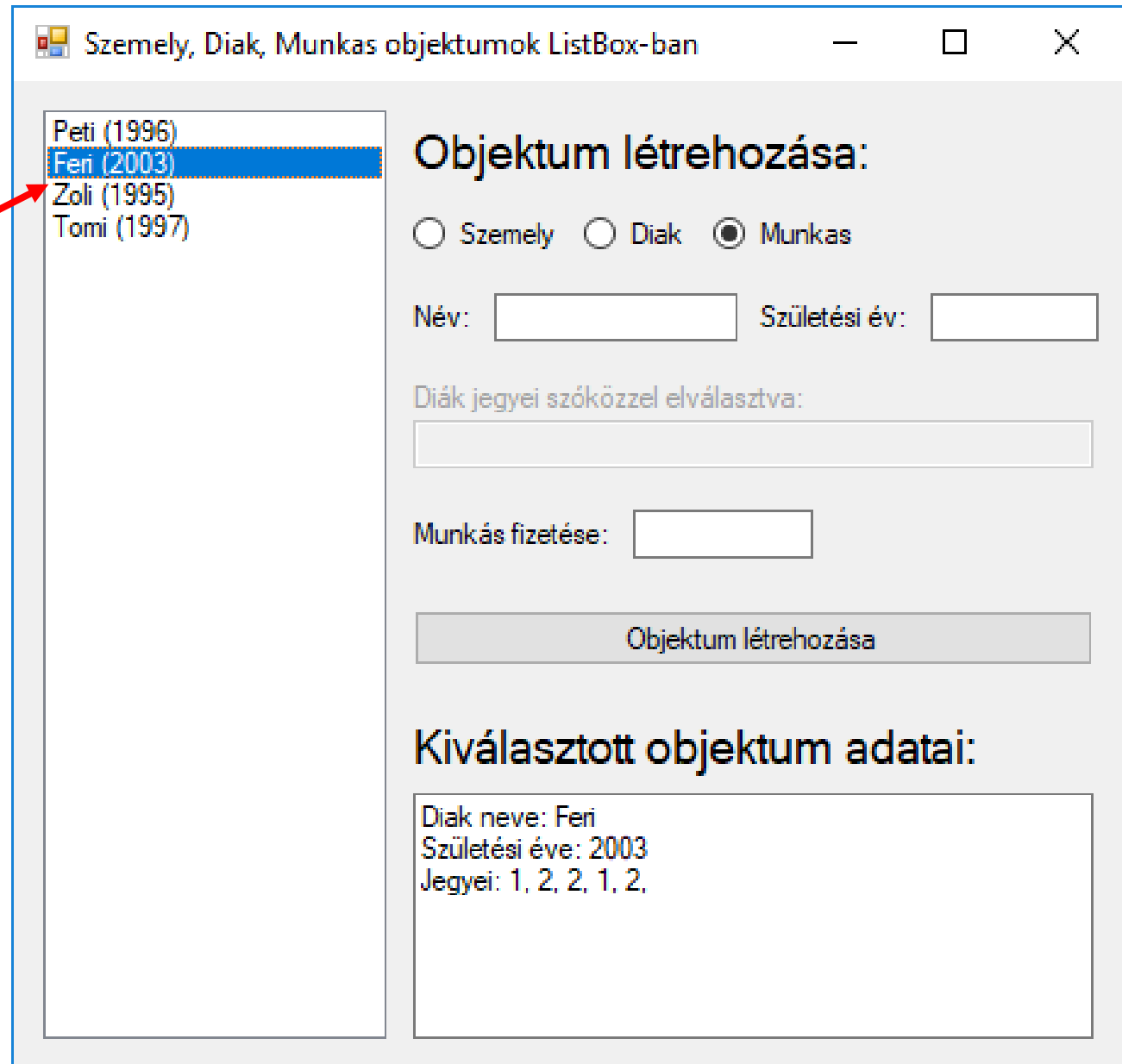
private void button1_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    for (int i=0; i<a.Length; i++)
    {
        textBox1.AppendText(a[i] + "\n");
    }
}
```

Peti (1996) Feri (2003) Zoli (1995)

Objektumok tárolása ListBox-ban:

018 Objektumok ListBox-ban

a ListBox-ban az
objektumok ToString()
metódusa által
visszaadott karakterlánc
jelenik meg



Szemely, Diak, Munkas objektumok ListBox-ban

Objektum létrehozása:

☐ Szemely ☐ Diak ☒ Munkas

Név: Születési év:

Diák jegyei szóközzel elválasztva:

Munkás fizetése:

Objektum létrehozása

Kiválasztott objektum adatai:

Diak neve: Feri
Születési éve: 2003
Jegyei: 1, 2, 2, 1, 2,

```
private void button1_Click(object sender, EventArgs e)
```

```
{
```

```
    Szemely x;
```

```
    if (radioButton1.Checked)
```

```
    {    // Szemely letrehozasa
```

```
        x = new Szemely(textBox1.Text, Convert.ToInt32(textBox2.Text));
```

```
    } else if (radioButton2.Checked)
```

```
    {    // Diak letrehozasa
```

```
        string[] jegyekString = textBox3.Text.Split(' ');
```

```
        int[] jegyekInt = new int[jegyekString.Length];
```

```
        for (int i = 0; i < jegyekString.Length; i++)
```

```
        {
```

```
            jegyekInt[i] = Convert.ToInt32(jegyekString[i]);
```

```
        }
```

```
        x = new Diak(textBox1.Text, Convert.ToInt32(textBox2.Text),  
                    jegyekInt);
```

```
    } else
```

```
    {    // Munkas letrehozasa
```

```
        x = new Munkas(textBox1.Text, Convert.ToInt32(textBox2.Text),  
                        Convert.ToDouble(textBox4.Text));
```

```
    }
```

```
    // Objektum hozzaadasa a ListBox elemeihez
```

```
    listBox1.Items.Add(x);
```

```
    // TextBox-okban levo szoveget torlese
```

```
    textBox1.Text = "";
```

```
    textBox2.Text = "";
```

```
    textBox3.Text = "";
```

```
    textBox4.Text = "";
```

```
}
```

objektum
létrehozása

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    textBox5.Clear();
    Szemely x = (Szemely)listBox1.SelectedItem;
    if (x is Diak)
    {
        textBox5.AppendText("Diak neve: " + x.Nev + "\n");
        textBox5.AppendText("Születési éve: " + x.SzuletesiEv + "\n");
        textBox5.AppendText("Jegyei: ");
        foreach (int j in (x as Diak).OsszesJegy())
        {
            textBox5.AppendText(j + ", ");
        }
        textBox5.AppendText("\n");
    } else if (x is Munkas)
    {
        textBox5.AppendText("Munkas neve: " + x.Nev + "\n");
        textBox5.AppendText("Születési éve: " + x.SzuletesiEv + "\n");
        textBox5.AppendText("Fizetése: " + (x as Munkas).GetFizetes() + "\n");
    } else
    {
        textBox5.AppendText("Szemely neve: " + x.Nev + "\n");
        textBox5.AppendText("Születési éve: " + x.SzuletesiEv + "\n");
    }
}
```

kiválasztott objektum
adatainak kiírása

Partial class: egy osztály több állományban található.

Munkas.cs állomány:

019 OOP Partial class

```
namespace _019_OOP_Partial_class
{
    public partial class Munkas : Szemely
    {
        private double _fizetes;

        public Munkas(string aNev, int aSzuletesiEv, double aFizetes) :
            base(aNev, aSzuletesiEv)
        {
            _fizetes = aFizetes;
        }

        public double GetFizetes()
        {
            return _fizetes;
        }
    }
}
```

Munkas2.cs állomány:

```
using System;
```

```
namespace _019_OOP_Partial_class
```

```
{
```

```
public partial class Munkas
```

```
{
```

```
public void FizetesEmeles(int szazalek)
```

```
{
```

```
    _fizetes = _fizetes + _fizetes * (szazalek / 100.0);
```

```
    _fizetes = Math.Round(_fizetes, 2);
```

```
}
```

```
public void FizetesCsokkentek(int szazalek)
```

```
{
```

```
    _fizetes = _fizetes - _fizetes * (szazalek / 100.0);
```

```
    _fizetes = Math.Round(_fizetes, 2);
```

```
}
```

```
}
```

```
}
```

az osztályt elég az egyik részben
megadni (ha mindkét részben
megadjuk, ugyanannak kell lennie)

Metódus túlterhelés: ha egy osztályban több ugyanolyan nevű metódus szerepel különböző paraméterekkel.

Szemely.cs állományban:

020 OOP Metodus tulterheles

...

```
public int Eletkor()  
{  
    return DateTime.Now.Year - SzuletesiEv;  
}
```

```
public int Eletkor(int evszam)  
{  
    return evszam - SzuletesiEv;  
}
```

...

Konstruktor túlterhelés: ha egy osztályban több ugyanolyan nevű konstruktor szerepel különböző paraméterekkel.

Diak.cs állományban:

020 OOP Metodus túlterheles

...

```
public Diak(string aNev, int aSzuletesiEv, int[] aJegyek) :  
    base(aNev, aSzuletesiEv)  
{  
    for (int i = 0; i < aJegyek.Length; i++)  
    {  
        _jegyek[i] = aJegyek[i];  
    }  
    _jegyekSzama = aJegyek.Length;  
}
```

```
public Diak(string aNev, int aSzuletesiEv) :  
    base(aNev, aSzuletesiEv)  
{  
    _jegyekSzama = 0;  
}
```

...

Összefoglalás:

- Az ősosztály helyettesíthető a belőle leszármaztatott osztállyal:
Szemely x = new Munkas(„Zoli”, 1995, 590);
- Statikus típus („Szemely”) és dinamikus típus („Munkas”)
- Statikus kötés („new” módosító) és dinamikus kötés („virtual” és „override” módosítók)
- Osztályok hierarchiája, „Object” osztály
- „ToString()” metódus
- Objektumok tárolása ListBox-ban
- Partial class
- Metódus túlterhelés, konstruktor túlterhelés