Eötvös Loránd Tudományegyetem
Informatikai Kar

Eseményvezérelt alkalmazások

---

2. előadás

Windows Forms alapismeretek, eseményvezérlés

Cserép Máté
mcserep@inf.elte.hu
https://mcserep.web.elte.hu

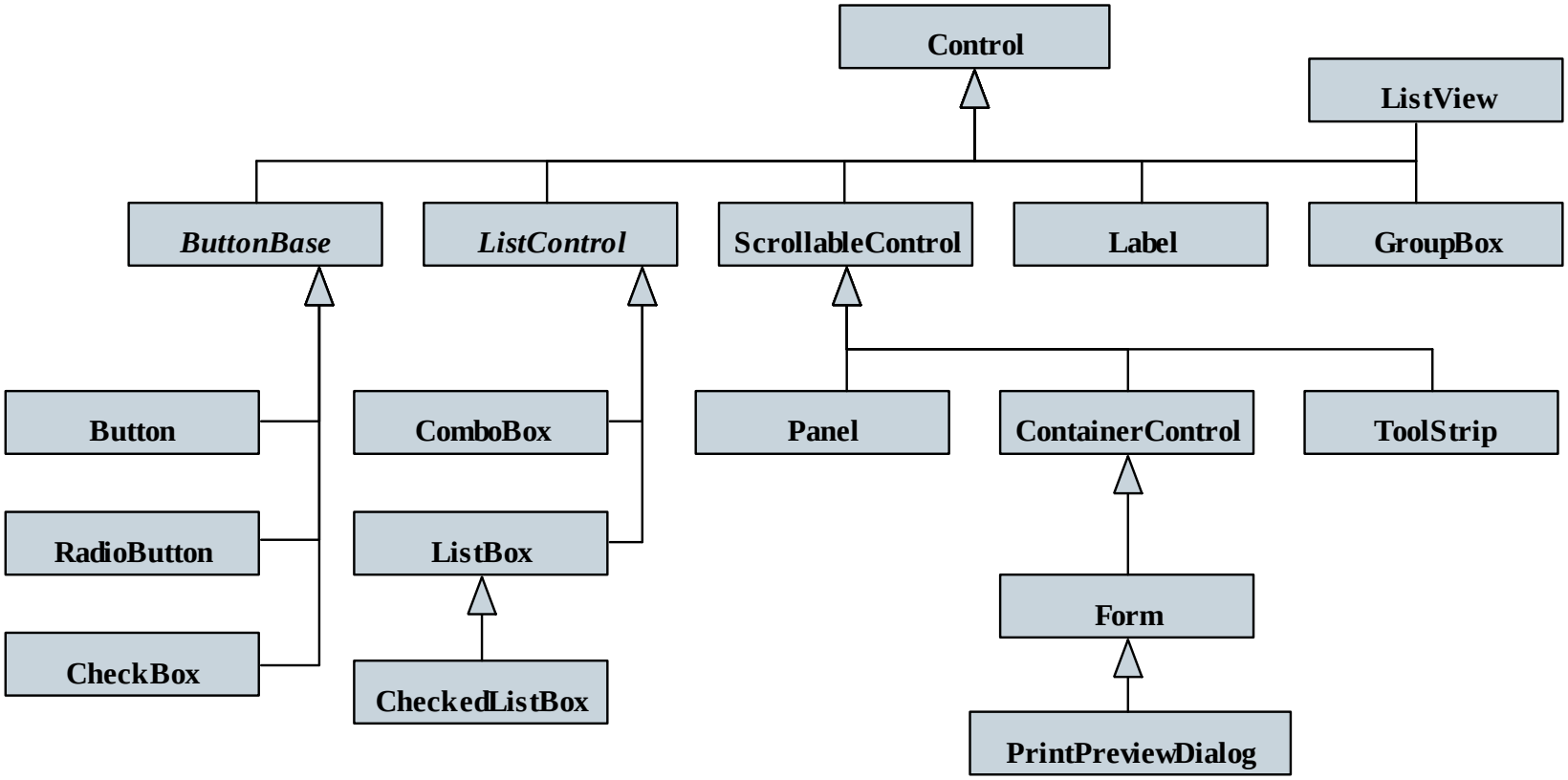
Windows Forms alapismeretek

A felület

- A *Windows Forms* (*WinForms*) a .NET keretrendszer első grafikus felülete
  - raszteres grafikára (GDI+) épül, és teljes mértékben processzor által vezérelt
  - a grafikus felület előállításához a *Microsoft Visual Studio* biztosít egy felülettervező eszközt, amivel grafikusan készítjük el a felületet, a hozzá tartozó kódot pedig legenerálja az eszköz
  - alapvetően vezérlőkből épül fel, amelyek a **`System.Windows.Forms`** névtérben helyezkednek el, pl.
    - gombok (**`Button`**, **`RadioButton`**, **`CheckBox`**, …),
    - beviteli mezők (**`TextBox`**,**`ComboBox`**, **`ListBox`**, …),
    - dialógusablakok (**`MessageBox`**, **`OpenFileDialog`**, …)

Windows Forms alapismeretek

Vezérlők



Windows Forms alapismeretek

Vezérlők tulajdonságai

- A vezérlőket tulajdonságaik segítségével szerkeszthetjük, pl.:
  - pozícionálás és méretezés (**`Location`**, **`Size`**, **`Anchor`**, **`AutoSize`**, **`Dock`**)
  - engedélyezettség (**`Enabled`**), fókusz (**`Focused`**)
  - felirat (**`Text`**), szöveget tartalmazó elemekben
  - színezés (**`ForeColor`**, **`BackColor`**), amelyek a **`Color`** osztály segítségével állíthatunk be tetszőleges RGB kombinációra, vagy fix értékre (pl. **`Color.Red`**)

```
Label myLabel = new Label(); // új címke
myLabel.Location = new Point(6, 18); // pozíció
myLabel.ForeColor = Color.Blue; // szövegszín
myLabel.Text = "valami felirat"; // felirat
```

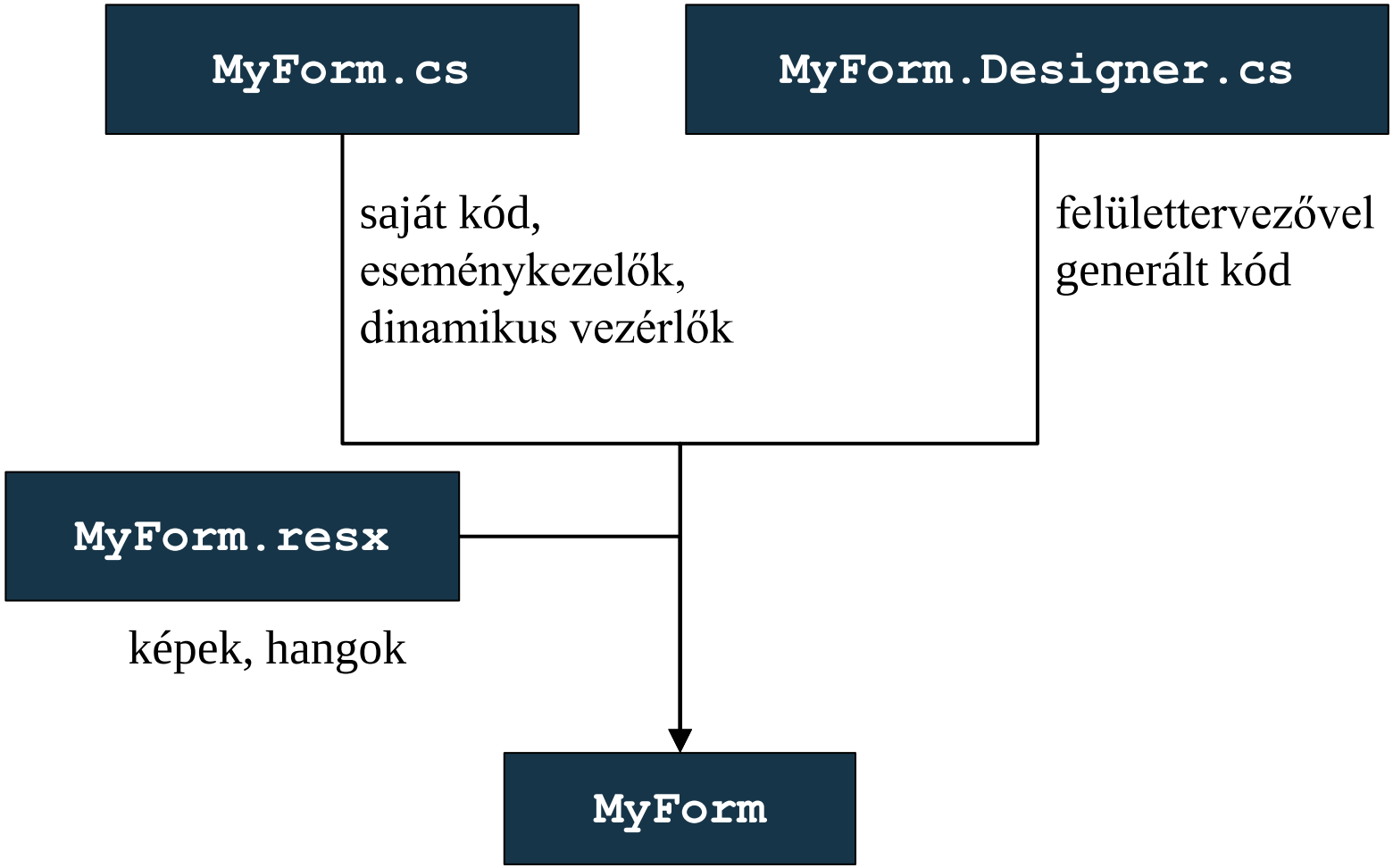
Windows Forms alapismeretek

Ablakok

- Az ablakok osztályok (a **`Form`** osztály leszármazottai), amelyben definiálhatjuk az ablak vezérlőit, és azok viselkedését
  - speciális tulajdonságokkal szabályozhatjuk a megjelenést, pl. vezérlő eszköztár (**`ControlBox`**, **`MinimizeBox`**, **`MaximizeBox`**), menü (**`Menu`**), kezdőpozíció (**`StartPosition`**)
  - az ablakok általában parciális (**`partial`**) osztályok, amelyek két fájlban helyezkednek el:
    - egy a programozott (***`<osztálynév>`*`.cs`**),
    - egy a generált kódot (***`<osztálynév>`*`.Designer.cs`**) tartalmazza, pontosabban az **`InitializeComponent()`** metódus, amelyet az osztály konstruktora futtat (így a vezérlők csak ennek lefutását követően érhetőek el)

Windows Forms alapismeretek

Ablakok



Windows Forms alapismeretek

Ablakok

- Pl. (**MyForm.cs**):

```
namespace MyFormsApplication
{
    partial class MyForm : Form {
        // parciális ablak osztály
        public MyForm() // konstruktor
        {
            InitializeComponent();
              // generált vezérlők létrehozása

            … // további tevékenységek
        }
    }
}
```

Windows Forms alapismeretek

Ablakok

- Pl. (`MyForm.Designer.cs`):

```
namespace MyFormsApplication
{
    partial class MyForm {
        // parciális ablak osztály másik része

        public void Dispose() { … }
            // ablak megsemmisítése
        public void InitializeComponent(){ … }
            // vezérlők inicializálása

        // … vezérlők mezői
    }
}
```

Windows Forms alapismeretek

Dialógusablakok

- A dialógusablakok egyszerű funkciókat megvalósító ablakok, a legegyszerűbb az előugró üzenet (**`MessageBox`**)
  - a statikus **`Show(…)`** művelettel használható, amely paraméterezhető (pl. üzenet, gombok, ikon, …)
  - a művelet visszatérési értéke **`DialogResult`**, így lekérdezhető, milyen gombot használt a felhasználó
  - pl.:

```
MessageBox.Show("Really quit?",
    "My Application", // cím
    MessageBoxButtons.YesNo, // gombok
    MessageBoxIcon.Question); // ikon
```

Windows Forms alapismeretek

Dialógusablakok

- A további dialógusablakok megegyeznek az operációs rendszerben fellelhető ablakokkal, pl.:
  - fájl megnyitó (**`OpenFileDialog`**), fájl mentő (**`SaveFileDialog`**), könyvtárböngésző (**`FolderBrowserDialog`**)
  - betűtípus-választó (**`FontDialog`**), színválasztó (**`ColorDialog`**)
  - nyomtatási beállítások (**`PrintDialog`**), előnézet (**`PrintPreviewDialog`**), oldalbeállítás (**`PageSetupDialog`**)
- További dialógusablakok (pl. szövegbeviteli mező) egyedileg készíthetőek

Windows Forms alapismeretek

Dialógusablakok

- Pl. :

```
SaveFileDialog dialog = new SaveFileDialog();
   // fájl mentő dialógus
dialog.Title = "Save file"; // cím
dialog.Filter =
   "txt files (*.txt)|*.txt|All files (*.*)|*.*";
   // szűrés a megjelenített tartalomra
if (dialog.ShowDialog() == DialogResult.OK) {
   // ha OK-val zárták le az ablakot
   StreamWriter writer =
      new StreamWriter(dialog.FileName);
   // a megadott fájlnévre mentünk
   …
}
```

Windows Forms alapismeretek

Alkalmazás osztályok

- A grafikus felületű alkalmazásokat egy *alkalmazásnak* (**`Application`**) kell vezérelnie
  - statikus osztály, a főprogramban használjuk
  - legfőbb művelete a futtatás (**`Run`**), amely paraméterben megkapja az első indítandó képernyő objektumát, illetve lehetőséget ad a kilépésre is (**`Exit`**)
  - ezen felül alkalmas a környezet beállítására (**`SetHighDpiMode`**, **`EnableVisualStyles`**, **`UseWaitCursor`**, …), valamint információgyűjtésre (**`StartupPath`**, **`OpenForms`**, **`ProductName`**, …)
  - eseményeivel követhetjük a programfutást (**`ApplicationExit`**, **`Idle`**)

Windows Forms alapismeretek

Alkalmazás osztályok

- Pl. (**Program.cs**):

```
namespace MyFormsApplication
{
    class Program
    {
        [STAThread]
        static void Main() // főprogram
        {
            ApplicationConfiguration.Initialize();
            Application.Run(new MyForm());
        // alkalmazás indítása a megadott ablakkal
        }
    }
}
```

Események és eseménykezelés

- Az *akció* az adott alkalmazás aktuális tevékenységétől függetlenül bekövetkező történés, amelyet kezdeményezhet a felhasználó (billentyűzet, egér, érintőképernyő, stb.), az alkalmazás egyik objektuma (pl. időzítő tikkelése), vagy egy másik alkalmazás.

- Az *esemény* (*event*) az alkalmazásban objektumként megjelenő akció.
  - Egy billentyű leütés akciójából a lenyomás és a felengedés eseménye lesz.
  - Az egér jobb fülével történő kattintás akcióból egy „egérgomb-lenyomás” és egy „egérgomb-felengedés” esemény születik.

Windows Forms alapismeretek

Események és eseménykezelés

- Egy eseményre az alkalmazás valamelyik vezérlő objektuma reagálhat úgy, hogy végrehajt valamilyen tevékenységet, ezt nevezzük az *eseménykezelőnek*.
  - Ugyanazt az esemény akár több vezérlő objektum is megkapja, és többen is reagálhatnak rá (kezelhetik).
  - Ugyanazt a reakciót több különböző esemény is kiválthatja (pl. egy fókuszban levő nyomógomb feletti egér-kattintás, vagy az <*enter*> leütése ugyanazon gomb megnyomását okozza).

Események és eseménykezelés

- A C# nyelvi szinten valósítja meg az eseménykezelést, amelyhez eseményeket (**`event`**) és delegáltakat (**`delegate`**) használ
  - az eseménykezelő egy szabványos metódus, de konvenció szerint két paramétere van, a küldő objektum (**`object sender`**), és az eseménytulajdonságok (**`EventArgs e`**), amelyek leszármazottai hordozhatnak speciális értéket
  - a delegált szabja meg az eseménytulajdonságok (**`EventArgs`**) típusát
    - az alapértelmezett delegált az **`EventHandler`**
    - lehet delegáltakat létrehozni, vagy sablont használni más tulajdonságokhoz

Windows Forms alapismeretek

Események és eseménykezelés

- Az eseménykezelő hozzárendelésekor az eseménykezelő nevét kell megadnunk:

```
<objektumnév>.<eseménynév>
    += new EventHandler(<metódusnév>);
```

  - a **+=** operátor lehetővé teszi, hogy egy eseményhez több eseménykezelőt is hozzárendeljünk
  - a társításban bármely objektum eseményét rendelhetjük bármely, azonos szintaktikájú eseménykelőhöz
  - a **-=** operátor segítségével tudjuk bontatni a kapcsolatot
- Pl.:

```
class EventClass {
   public event EventHandler MyEvent; // esemény
}
```

- Pl.:

```
class HandlerClass {
   private EventClass ec;

   public HandlerClass(){
      ec = new EventClass();
      ec.MyEvent +=
         new EventHandler(MyEventHandler);
      // eseménykezelő társítás
   }
   private void MyEventHandler(object? sender,
                               EventArgs e){ … }
      // eseménykezelő metódus
}
```

Windows Forms alapismeretek

Vezérlők eseményei

- A vezérlők számos eseménnyel rendelkeznek, több csoportban:
  - egér és billentyűzet tevékenységek (**`Click`**, **`MouseClick`**, **`MouseHover`**, **`KeyDown`**, **`KeyUp`**, …)
  - vezérlőállapot megváltozása (**`Validating`**, **`Validated`**, **`Resize`**, **`Paint`**, **`GotFocus`**, …)
  - tulajdonságok megváltozása (**`BackColorChanged`**, **`TabIndexChanged`** , **`TextChanged`**, **`SizeChanged`**, …)
- Bizonyos események csak akkor váltódnak ki, ha a vezérlő fókuszban van (**`Focus()`**), pl. billentyűzetesemények
  - ugyanakkor a billentyűzet lekezelhető az ablak szintjén is

Windows Forms alapismeretek

Vezérlők eseményei

- Pl.:

```
Button b = new Button();
b.Click += new EventHandler(B_Click); // társítás
b.MouseDoubleClick +=
    new MouseEventHandler(B_DClick); // társítás
…
void B_Click(object? sender, EventArgs e) { … }
    // eseménykezelő
…
void B_DClick(object? sender, MouseEventArgs e) {
    // speciális eseményargumentum, amelytől
    // lekérdezhető az egérgomb (Button) és a
    // pozíció (Location)
}
```

Windows Forms alapismeretek

Példa

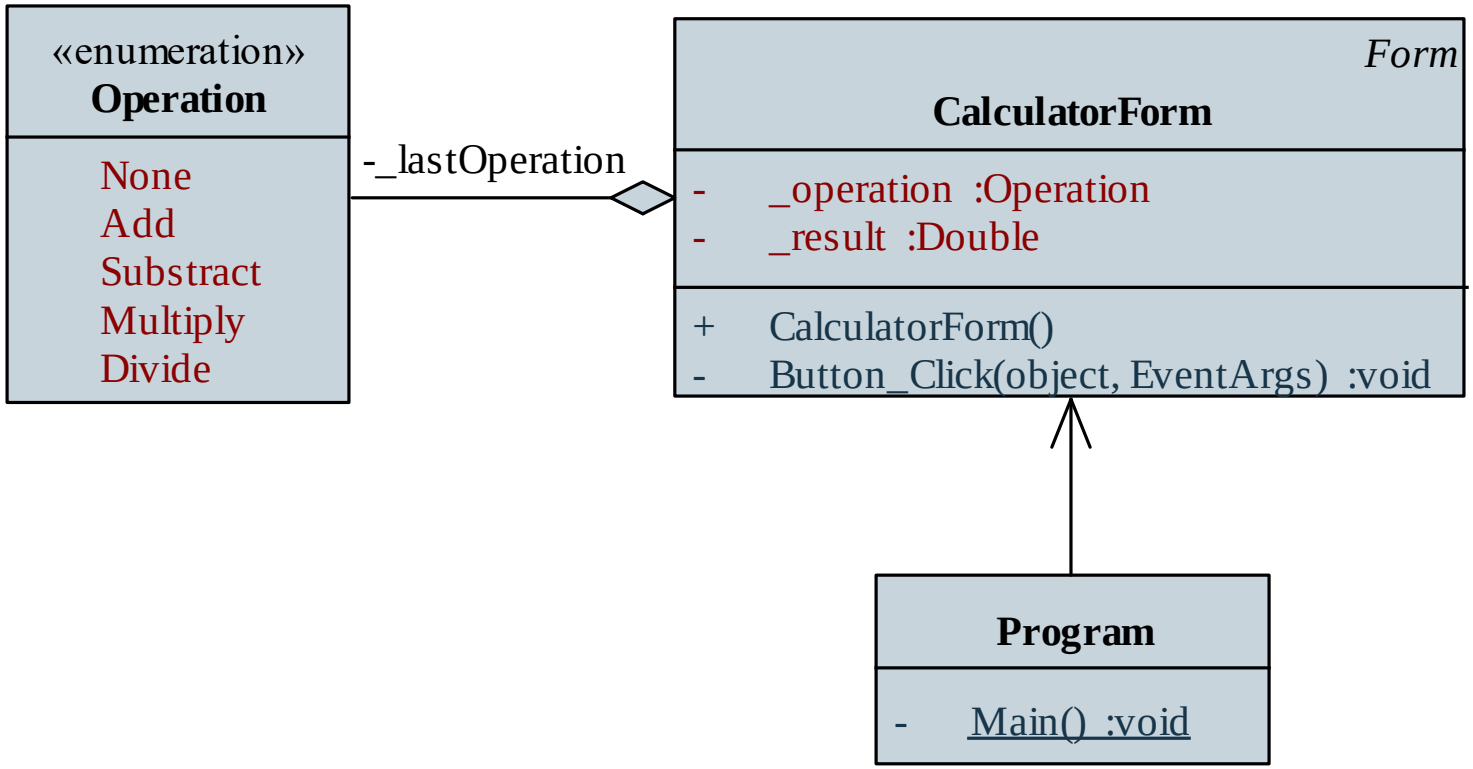
Feladat: Készítsünk egy egyszerű számológépet, amellyel a négy alapműveletet végezhetjük el, illetve láthatjuk korábbi műveleteinket is.

- az alkalmazás felületét a felülettervezővel készítjük el, elhelyezünk 5 gombot (**`Button`**), egy szövegbeviteli mezőt (**`TextBox`**), valamint egy listát (**`ListBox`**)
- az ablak osztályban (**`CalculatorForm`**) létrehozunk egy eseménykezelőt (**`Button_Click`**) a gombokra, amely a megfelelő műveleteket végzi el
- egy felsorolási típussal (**`Operation`**) tároljuk el a műveletet
- ellenőrizzük kivételkezeléssel, hogy a bevitt érték megfelelő-e

Windows Forms alapismeretek

Példa

Tervezés:



Windows Forms alapismeretek

Példa

Megvalósítás (`CalculatorForm.cs`*):*

```
public partial class CalculatorForm : Form {
    …
    // egy közös eseménykezelő az összes gombnak
    private void Button_Click(object? sender,
                              EventArgs e){
        try {
            …
            // minden esetben:
            _firstNumber =
                Double.Parse(_textNumber.Text);
                // eltároljuk az első operandust
```

Windows Forms alapismeretek

Példa

Megvalósítás (`CalculatorForm.cs`*):*

```
            if (sender is Button button)
                switch (button.Text) {
                    // megvizsgáljuk, milyen az
                    // eseményt kiváltó gomb felirata,
                    // így eldönthetjük, melyik gombot
                    // nyomták le
                    …
        }
        catch (OverflowException) {
            MessageBox.Show("Your input has too many
                digits!", "Calculation Error",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        …
```

Windows Forms alapismeretek

Ablakok használata

- Ablakok megnyitására két lehetőségünk van:
  - a **`Show()`** művelet megnyitja az ablakot, de utána tovább fut a megnyitó ablak kódja
  - a **`ShowDialog()`** művelet dialógusablakként nyitja meg, ekkor a megnyitó ablak blokkolódik, és csak az új ablak bezárása után lehet bármely más tevékenységet végezni
  - utóbbi esetben kaphatunk eredményt (**`DialogResult`**) az ablaktól a lezárást illetően (pl. **`None`**, **`OK`**, **`Cancel`**, **`Yes`**, …), amelyet lekérdezhetünk, pl.:
    ```
    if (myForm.ShowDialog() == DialogResult.Yes) …
    ```
- Ablak bezárása a **`Close()`** művelettel történik

Windows Forms alapismeretek

Időzítő

- Az időzítő kezelést egyfelől szálak segítségével, másfelől a **`Timer`** osztályon keresztül vehetjük igénybe
  - lehetőségünk van indításra (**`Start`**), leállításra (**`Stop`**), állapotlekérdezésre (**`Enabled`**), valamint az intervallum (**`Interval`**) beállítására, az idő eltelésekor a **`Tick`** esemény váltódik ki
  - pl.:

```
Timer myTimer = new Timer(); // időzítő
myTimer.Interval = 1000;
   // 1 másodpercenként váltódik ki az esemény
myTimer.Tick += new EventHandler(Timer_Tick);
   // eseménykezelő társítás
myTimer.Start(); // indítás
```

Windows Forms alapismeretek

Példa

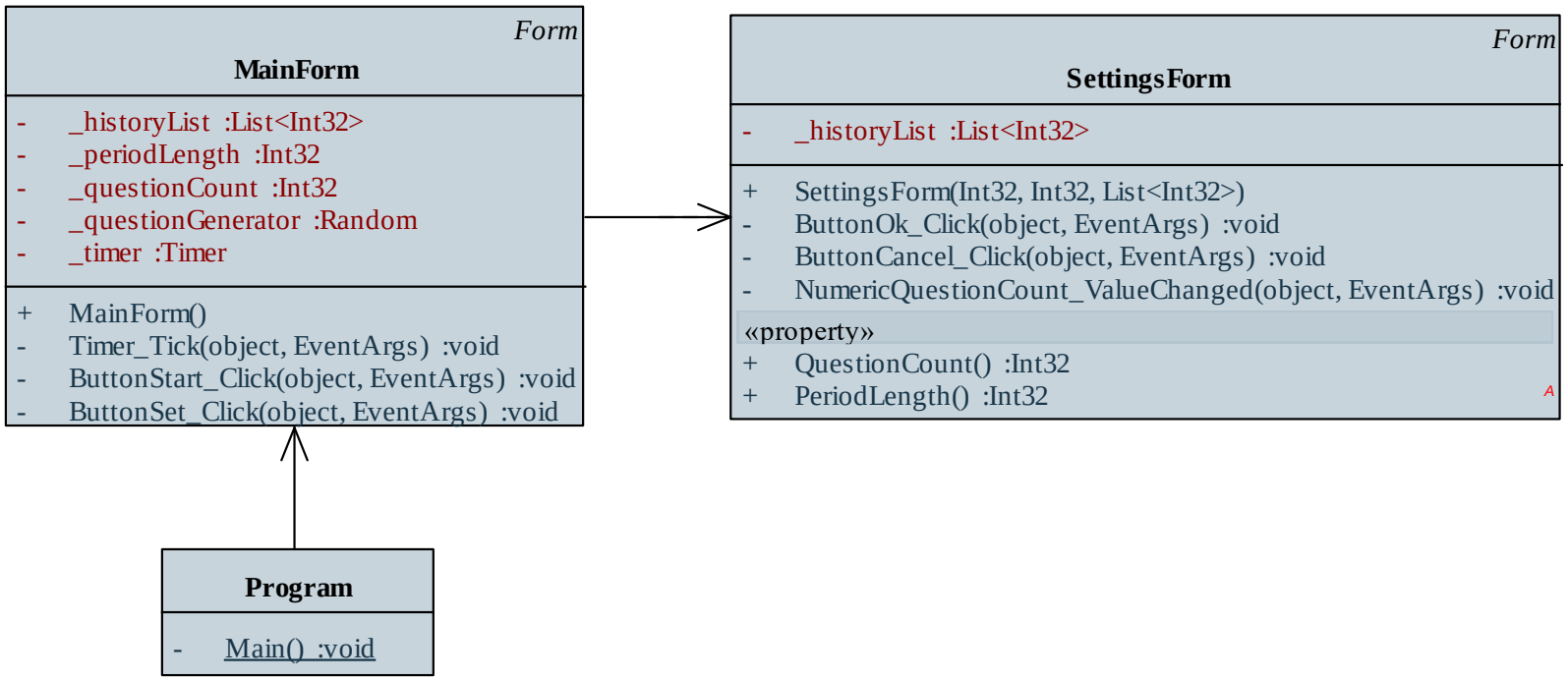
Feladat: Készítsünk egy vizsgatétel generáló alkalmazást, amely ügyel arra, hogy a vizsgázók közül ketten ne kapják ugyanazt a tételt.

- a főablakban két gombot (*Start/Stop*, *Beállít*), valamint egy szövegmezőt helyezünk el, a generálást időzítővel (**`Timer`**) valósítjuk meg, a generált számokat elmentjük egy listába az ellenőrzéshez
- egy segédablakban két számbeállító (**`NumericUpDown`**) segítségével állítjuk be a tételek számát és a bent lévő hallgatók számát
- egy kijelölhető lista (**`CheckedListBox`**) segítségével ellenőrizhetjük és korrigálhatjuk a kiadott tételszámokat

Windows Forms alapismeretek

Példa

Tervezés:



Windows Forms alapismeretek

Példa

Megvalósítás (`MainForm.cs`*):*

```
void Timer_Tick(object? sender, EventArgs e) {
   Int32 number = _questionGenerator.Next(1,
      _questionCount + 1);
      // új szám generálása 1 és a tételszám
      // között
   while (_historyList.Contains(number))
      // ha a szám szerepel a korábbiak között
      number = _questionGenerator.Next(1,
         _questionCount + 1);
         // akkor új generálása

   _textNumber.Text = number.ToString();
}
```

Példa

Megvalósítás (`MainForm.cs`):

```
void ButtonSet_Click(object? sender, EventArgs e)
{
    SettingsForm f = new SettingsForm(
        _questionCount, _periodLength,
        _historyList);
        // dialógusablak létrehozása paraméterekkel

    if (f.ShowDialog() == DialogResult.OK) {
        // dialógusablak megjelenítése
        _questionCount = f.QuestionCount;
            // elmentjük az új értékeket
        _periodLength = f.PeriodLength;
}
```