Eötvös Loránd Tudományegyetem
Természettudományi Kar

Alkalmazott Modul III

7. gyakorlat

Eseményvezérelt programozás: grafikus felület és eseménykezelés

© 2011.11.08. Giachetta Roberto
groberto@inf.elte.hu
http://people.inf.elte.hu/groberto

Grafikus felületű alkalmazások

Működése

- *Grafikus felületű alkalmazás*nak nevezzük azt a programot, amely 2D-s interaktív felhasználó felületen (*GUI*, *Graphical User Interface*) keresztül kommunikál a felhasználóval
  - ez lényegesen gazdagabb interakciót tesz lehetővé, mint a konzol felület
  - a felhasználó felület egy, vagy több ablakból (*form/window*) áll, amelyek vezérlőket (*control*) tartalmaznak (pl.: nyomógombok, listák, menük, rajzpanelek, ...), és a vezérlők további vezérlőkből állhatnak
  - mindig van egy aktív ablak, és egy aktív vezérlő, amelyen épp munkát végezhet a felhasználó (ezen van a *fókusz*), és lehet váltani fókuszt a vezérlők között

Grafikus felületű alkalmazások

Felépülése

- A grafikus felület elemei viselkedéssel és adatokkal is rendelkeznek, ezek legkönnyebb őket *objektumorientált programozás* segítségével leírni
  - minden elem a képernyőn (maga az ablak is) objektum lesz, és amelyet attribútumain (mezőin) keresztül konfigurálhatunk (pl. pozíció, méret, szín, szöveg, ...)
  - az elemek sok közös tulajdonsággal rendelkeznek, ezért célszerű öröklődés segítségével leírni őket, így a közös tulajdonságok összefoghatóak az általános osztályba
  - az öröklődéshez egy teljes hierarchiát építünk fel, mivel több szinten is lehetnek egyezések, pl. vezérlőkön belül gombok, azon belül nyomógombok csoportjait tudjuk megkülönböztetni

Grafikus felületű alkalmazások

Felépülése

- az ablakok olyan speciális vezérlők, amelyek egyedi viselkedéssel rendelkeznek, és tartalmazhatnak más vezérlőket, mivel ez minden egyes ablakra más, a különböző ablakfajtákat önálló osztályok segítségével valósítjuk meg
- lehetőségünk van új, egyedi vezérlők definiálására specializációval

- Mivel a .NET Keretrendszer teljes származtatási hierarchiára épül, ez a filozófia jól bevezethető itt is, így a grafikus felület osztályai beépülnek az öröklődési fába
  - az ősosztály a vezérlő (`Control`) osztálya, amely definiálja a lényegi tulajdonságok jelentős részét

Grafikus felületű alkalmazások

Vezérlők

- A vezérlők létrehozásakor be kell állítani a mezőiket (pl. név, felirat, pozíció), ám ezt a konstruktoron keresztül tesszük, hanem külön
  - a név (**`Name`**) a későbbi azonosításra szolgál
  - a pozíciót és a méretet külön osztályok segítségével adhatjuk meg (**`Point`**, **`Size`**)
  - pl. egy címke esetén:

```
Label myLabel = new Label();
myLabel.Location = new Point(6, 18); // pozíció
myLabel.Name = "cimke"; // név
myLabel.Size = new Size(65, 13); // méret
myLabel.TabIndex = 1; // tabulátor index
myLabel.Text = "valami felirat"; // felirat
```

Grafikus felületű alkalmazások

Alkalmazás osztályok

- A grafikus felületű alkalmazásokat nem a főprogram vezérli, hanem egy külön vezérlő osztály, amelyet *alkalmazás osztálynak* (*application*) nevezünk
  - felügyeli a benne található összes grafikus vezérlőt, valamint a fókuszváltást
  - biztosítja az események feldolgozását, és megfelelő (aktív) objektumnak való továbbítását
  - beállítja az alkalmazástulajdonságokat (megjelenés, elérési útvonal, ...)

```
static void Main(){
   Application.EnableVisualStyles();
   Application.Run(new MainForm());
}
```

Grafikus felületű alkalmazások

A Visual Studio eszközei

- A *Microsoft Visual Studio* biztosít egy felülettervező eszközt, amivel grafikusan készítjük el a GUI-t, a hozzá tartozó kód pedig legenerálódik, így nem szükséges a teljes kódot megírnunk
  - egy eszköztárból (*ToolBox*) válogathatunk a vezérlők közül, és ezt grafikusan („drad and drop” módszerrel) helyezhetjük a felületre
  - a vezérlő tulajdonságait és műveleteit (*Properties*) külön állíthatjuk
  - bármikor válthatunk a kód és a tervező nézet között, és felváltva szerkeszthetjük a felületet

Grafikus felületű alkalmazások

Ablak osztályok

- A .NET 2.0 keretrendszerben a grafikus felületet a **`System.Windows.Forms`** névtér biztosítja, az ablakot a **`Form`** osztályból specializáljuk
  - az ablakaink úgynevezett parciális (**`partial`**) osztályok, mivel az osztály több fájlban helyezkedik el
  - a felülettervező által generált kód egy másik (úgynevezett tervező) fájlba kerül, így két fájl alkotja az ablak osztályát:
    `<osztálynév>.cs`

    `<osztálynév>.Designer.cs`
  - ezen felül az ablakhoz használt erőforrások (képek, ikonok, ...) egy erőforrás fájlban helyezkednek el (ez igazából egy XML leíró fájl):
    `<osztálynév>.resx`

Ablak osztályok

- az ablakban lévő vezérlők tulajdonságai az **`InitializeControls()`** metódusban kerülnek beállításra, amely meghívódik az osztály konstruktorában, ezt tetszőlegesen felülírhatjuk, illetve módosíthatjuk
  - minden itt történt módosítás hatással lesz a tervezőnézetre, a hibát okozó módosítást a tervező jelzi
  - a konstruktorban csak a hívás után írjunk egyéb kódot, előtte a vezérlők nincsenek inicializálva
- az ablakban létrehozott vezérlőket fel kell venni az ablak vezérlői (**`Controls`**) közé, pl.: **`Controls.Add(myLabel)`**
- emellett a tervező felülírja automatikusan a **`Dispose`** metódust, ami a vezérlők törlését végzi

Grafikus felületű alkalmazások

A .NET keretrendszerben

```
using System.Windows.Forms;
namespace <névtér>
{
  public partial class <ablak> : Form {
      // az ablak osztálya
      public <ablak>() { InitializeComponents(); }
  }
  static class Program {
      static void Main() {
          Application.Run(new <ablak>());
          // a főprogramban futtatunk egy ablakot
      }
  }
}
```

Eseményvezérlés

Grafikus felületen

- A programok vagy automatikusan végeznek tevékenységet, vagy a felhasználói tevékenységre reagálva futtatnak valamilyen algoritmust
  - a konzol felületű alkalmazásokban a felhasználó a billentyűzeten keresztül kommunikált a programmal
  - grafikus felületű alkalmazás esetén a billentyűzet mellett az egér is használható beviteli eszközként, illetve a program reagálhat más történésekre is (operációs rendszer üzenetei, más programok üzenetei)
- A program által lereagálhat történéseket magában a programban, vagy környezetében nevezzük *esemény*eknek (*event*, *signal*), a esemény függvényében a programot különböző viselkedésre késztethetjük

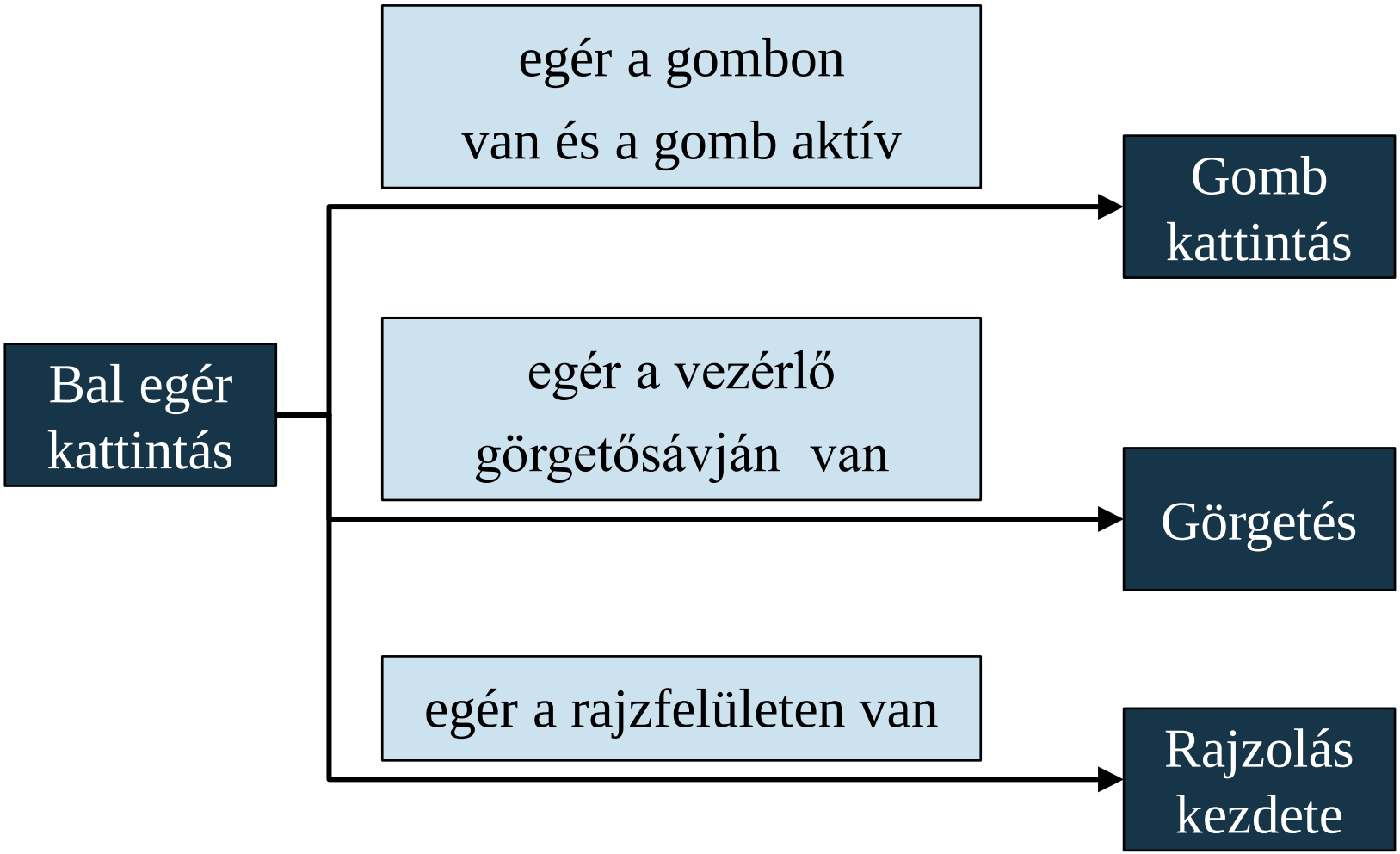
Eseményvezérlés

Körülmények

- Az eseményeknek van forrása, ami lehet beviteli eszköz, az operációs rendszer, vagy valamely (akár ugyanazon) program valamely objektuma
- Az esemény körülményei befolyásolhatják a kiváltott esemény típusát, ennek megfelelően összetett események is lehetnek, pl.:
  - gomb lenyomása, felengedése, elmozgatása
  - lista görgetése, elem behelyezése
  - ablak aktívvá válása, betöltése, bezárása, méretváltoztatás, fókuszváltás
- Általában az összetett események úgy valósulnak meg, hogy a program egy kiváltott eseményt lekezel, és további körülmények alapján új eseményt vált ki

Eseményvezérlés

Körülmények



Eseményvezérlés

Eseménykezelő társítások

- Az eseményre a program reakcióját egy *eseménykezelő alprogram* (*event handler*, *slot*) végzi
  - az eseménykezelőt az esemény típusához rendeljük

    ```
    button.Click += new Eventhandler(ButtonClick);
    ```

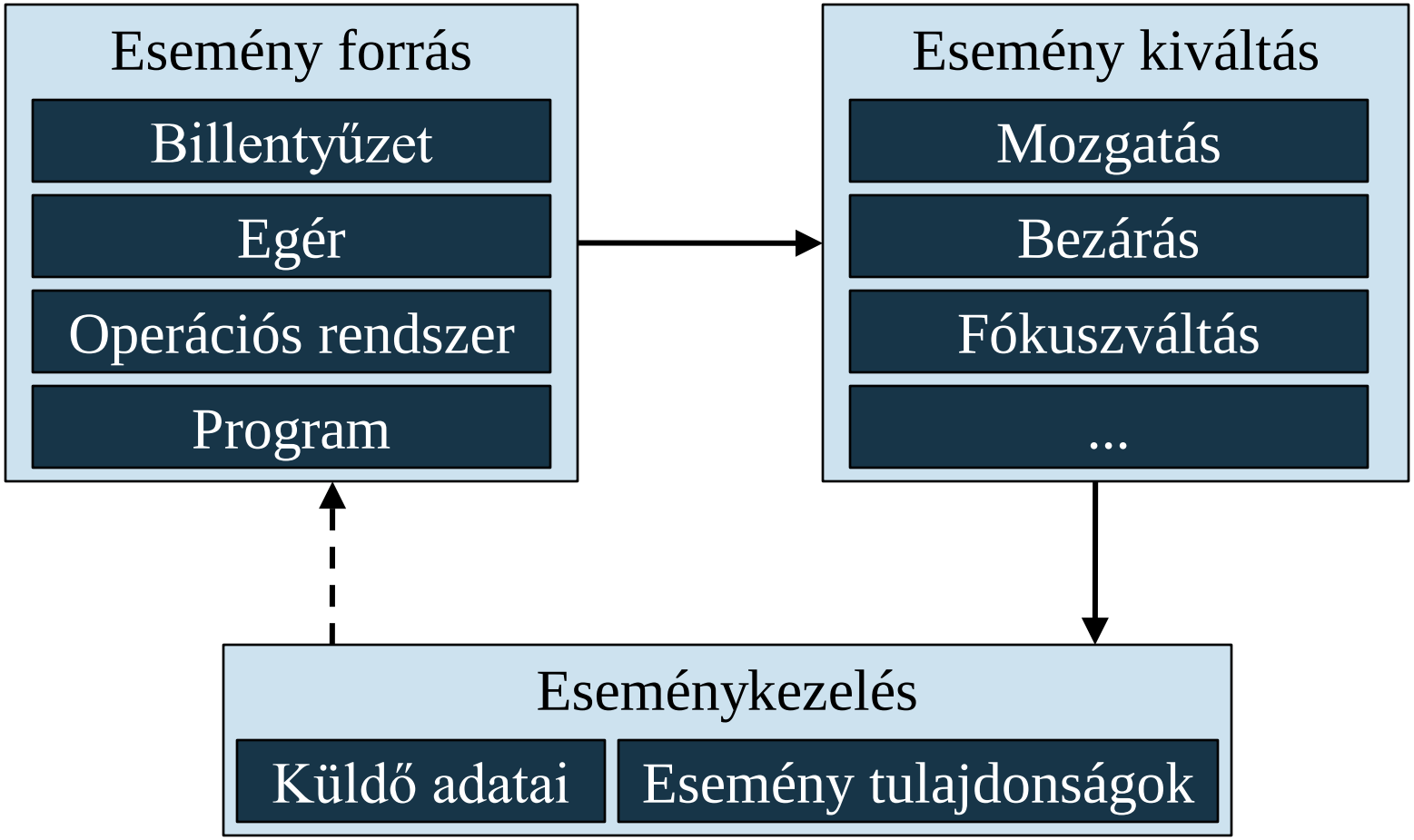
  - Az eseménykezelő paraméterezése kötött, de törzse tetszőleges lehet:

    ```
    void ButtonClick(object sender,
                     EventArgs e){...}
    ```

  - amely típusú eseményre a program nem rendelkezik eseménykezelővel, az kezeletlenül marad
- Az eseménykezelő mindig tisztában van az *esemény küldőjével* (*sender*), és annak állapotával, illetve az *esemény tulajdonságaival* (*event arguments*)

Eseményvezérlés

Mechanizmus



Eseményvezérlés

A .NET keretrendszerben

- A .NET-ben számos előre definiált eseménytípus adott, és egy eseménytípushoz tetszőlegesen sok eseménykezelőt rendelhetünk (persze rendszerint csak egyet fogunk)
  - lehet új eseménytípusokat is definiálni
  - lehet az ősben megírt eseménykezelőket felüldefiniálni
  - az eseménykezelőknek több típusa is van, az alaptípus az **`EventHandler`**
  - az eseménykezelőnek mindig két paramétere van, a küldő objektum (**`object sender`**), és az eseménytulajdonságok (**`EventArgs e`**)
- Az eseménykezelő hozzárendelése:

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

Eseményvezérlés

A .NET keretrendszerben

- Pl. egy gomb esetén:

```
Button gomb = new Button();
…
gomb.Text = "Egy gomb";
gomb.Click += new EventHandler(Gomb_Click);
    // egérkattintás esetén a Gomb_Click nevű
    // eseménykezelő metódus fut le
…

void Gomb_Click(object sender, EventArgs e){
    // eseménykezelő, a paramétereket most nem
    // használjuk, a gomb szövegét módosítjuk:
    gomb.Text = "Egy gomb kattintva lett";
}
```

Eseményvezérlés

A .NET keretrendszerben

- Több vezérlő többféle eseményéhez is rendelhetjük ugyanazt az eseménykezelőt, az a **`sender`** egyértelműen azonosítja azt az objektumot, amely az eseményt küldi
  - a küldő általános **`Object`** referenciával van megadva, de természetesen konvertálható a megfelelő típusnak
  - pl. két gombhoz ugyanazon eseménykezelő:

```
gomb1.Click += new EventHandler(Gomb_Click);
gomb2.Click += new EventHandler(Gomb_Click);
…
void Gomb_Click(object sender, EventArgs e){
    Button gomb = (Button)sender; // a küldő
    // mutatót átkonvertáljuk Button típusúvá
    gomb.Text = "Erre kattintottál";
}
```

Eseményvezérlés

Speciális eseménykezelők

- Az **`EventHandler`** leszármazottai speciálisabb eseménykezelők létrehozását teszik lehetővé, amelyek speciálisabb eseménytulajdonságokat tartalmaznak
  - pl. egéreseményeknél tudjuk az egér koordinátáját:

```
public MainForm(){
    InitializeComponents();
    this.MouseMove += // egérmozgatás eseménye
        new MouseEventHandler(Form_MouseMove);
}
void Form_MouseMove(object sender,
                    MouseEventArgs e){
    címke.Text = "Helyzet: " + e.X + ", " + e.Y;
    // e.X és e.Y az egérpozíciók
}
```

Grafikus felületű alkalmazások

Feladatok

1. Készítsünk egy olyan ablakot, amelyben egy piros négyzet (gomb) ugrál. Mindig az ablak közepén kezd, és egérkattintás hatására átugrik egy véletlenszerű pozícióba az ablakon.
2. Módosítsuk az előző programot úgy, hogy ne csak a gombra, hanem az egész ablakra kattintva mozogjon a gomb.
3. Módosítsuk az előző programot úgy, hogy a gomb színe váltakozzon véletlenszerűen az ugráskor (kezdetben legyen piros).
4. Módosítsuk az előző programot úgy, hogy ne véletlenszerűen változzon a szín, hanem a piros-fehér-zöld között váltakozzon ciklikusan.

Feladatok

5. Készítsünk egyszerű összeadó programot, amelyben a felhasználó megadhat két számot, és a program megjeleníti a két szám eredményét.
6. Készítsünk számológép alkalmazást, amely úgy működik, mint egy szokványos számológép, azaz tudja a négy alapműveletet, és van egy beviteli mező, amelyre megadhatjuk a következő értéket (azaz a művelet jobbértékét). Így nem a legutoljára kattintott művelet fog lefutni a programban, hanem az eggyel korábbi. Továbbá egy listába vegyük fel az eddigi számolásokat úgy, hogy az új érték a lista tetejére kerüljön.
7. Módosítsuk az előző programot úgy, hogy lehessen az állapotot (a lista tartalma, valamint az aktuális szám) kimenteni egy szöveges fájlba, majd később betölteni.

Feladatok

8. Készítsünk számolást tanító programot diákoknak. A felhasználó kiválaszthatja, milyen típusú műveletet szeretne gyakorolni (+, -, *, /). A program legenerálja a két értéket (0-100 között), aminek az eredményét a felhasználó megadja, majd utána a program ellenőrzi, hogy jó volt-e az eredmény.
9. Egészítsük ki az előző programot egy listával, amely tartalmazza az eddigi műveleteket, és sikerességüket. Továbbá számolja a program a sikerességi arányt is.