A programkód és az Unified Modeling Language (UML) osztálydiagramjának kapcsolata

Kiegészítés a Szoftvertechnológia I. tárgy UML témaköréhez

Készítette: Erdélyi Krisztina, PhD

Jelen segédlet az UML osztálydiagramjának mélyebb megértése céljából bemutatja az osztálydiagram és a hozzá tartozó kód kapcsolatát. A kód C# nyelven íródott, a diagramok Enterprise Architectben (EA) és Visual Studio 2012-ben (VS) készültek.

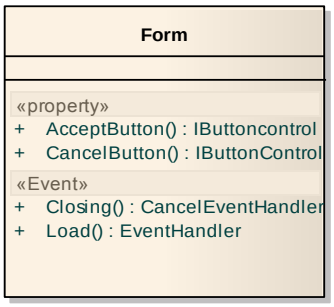
A Visual Studio diagramszerkesztőjét a .NET-es nyelvekhez fejlesztették ki. Alapelve, hogy a kód és a diagram ugyanannak a funkciónak a két megjelenítési formája. Éppen ezért:

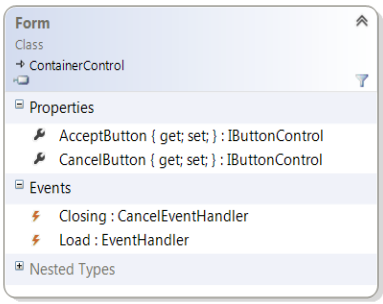
- a diagramot is a kód alapján generálja, a diagramot leíró fájlban csak annak megjelenítésére vonatkozó információt tárol,
- a diagramot és a kódot folyamatosan és automatikusan szinkronban tartja,
- csak azt jeleníti meg a diagramon, ami a kódból kiolvasható,
- igazodik a .NET-es nyelvek sajátosságaihoz.

Tehát a Visual Studio nem szabványos UML jelöléseket használ, de C#-ban a használata kényelmes. Az Enterprise Architect pontosan követi az UML jelölésrendszerét.

Sztereotípiák speciális nyelvi elemek megjelenítéséhez

Az UML jelölései nyelvfüggetlenek, ezért a modellezési nyelv a speciális programozási nyelvi elemek megjelenítéséről nem rendelkezik. A sztereotípiák nyújtanak lehetőséget az UML kiterjesztésére. Sztereotípiák segítségével speciális jellemzőket rendelhetünk egy-egy UML elemhez, így jelezve annak egyedi tulajdonságát. Például a C# nyelvben használt tulajdonság az UML-ben olyan metódus, amely <<Property>> sztereotípiával rendelkezik. (A sztereotípiákat mindig << és >> közé írjuk.) Az EA-ben ezt a megoldást látjuk. A sztereotípiák jelentenek megoldást az események megjelenítésére is. Az 1. ábra példát mutat tulajdonságok és események ábrázolására EA-ben (bal) és VS-ban (jobb).



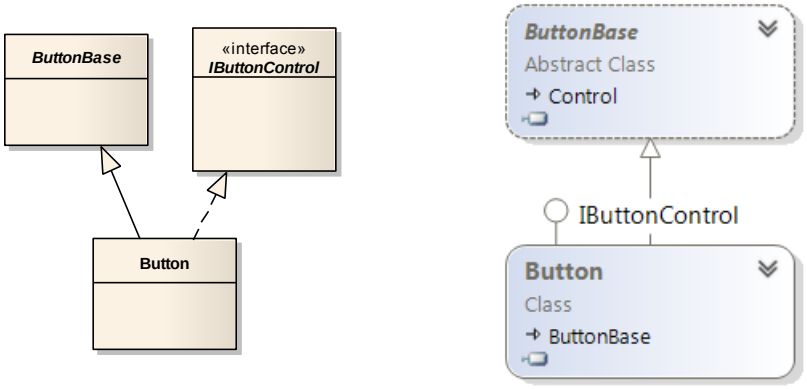



```
public class Form : ContainerControl
    {
      public IButtonControl AcceptButton { get; set; }
      public IButtonControl CancelButton { get; set; }
      public event CancelEventHandler Closing;
      public event EventHandler Load;
    }
```

1. ábra Tulajdonságok és események jelölése

Különböző típusok jelölése

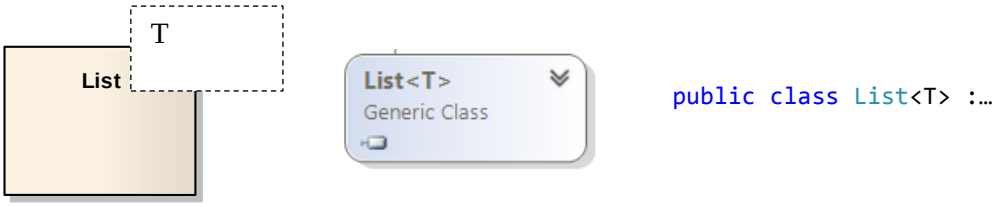
Absztrakt osztály jelölése UML-ben dőlt betű. ***Interfész*** jelölésére UML-ben használhatunk téglalapot <<interface>> sztereotípiával vagy kört (szárral a megvalósító osztályhoz csatlakoztatva). (2. ábra)




```
public abstract class ButtonBase : Control {…}
public class Button : ButtonBase, IButtonControl {…}
```

2. ábra Interfész és absztrakt osztály jelölése

Generikus típus jelölése az alábbiakban látható (3. ábra).

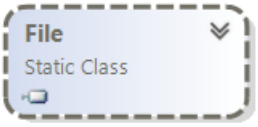


3. ábra Generikus típus jelölése

A ***struktúra***, a ***statikus osztály***, a ***felsorolás*** és a ***delegált*** jelölése szabványos UML-ben sztereotípiákkal oldható meg. A VS egyéb, megkülönböztető jegyeket használ:

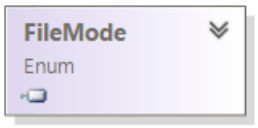
- struktúra: vastag keret (4. ábra),
- statikus osztály: szaggatott keret (5. ábra),
- felsorolás: lila háttér (6. ábra),
- delegált: rózsaszín háttér (7. ábra).

4. ábra Struktúra jelölése




```
public static class File {   }
```

5. ábra Statikus osztály jelölése




```
public enum FileMode {   }
```

6. ábra Felsorolástípus jelölése


```
public      delegate      void
EventHandler(object     sender,
EventArgs e);
```

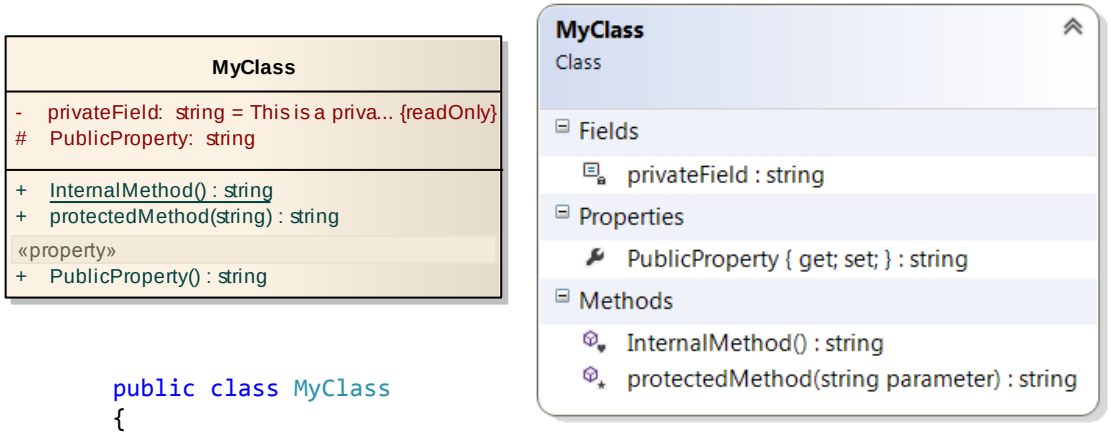
7. ábra Delegált jelölése

Tagok ábrázolása

Az osztályok ábrázolásánál a ***mezők*** és a ***metódusok*** része elkülönül. VS-ban ehhez jön még hozzá a tulajdonságok és az események része. (Az ***események*** és ***tulajdonságok*** jelölését lásd a Sztereotípiák speciális nyelvi elemek megjelenítéséhez című fejezetben.) Az UML lehetőséget biztosít az osztályok részletesebb vagy áttekintőbb megjelenítésére. Az EA-ban láthatósági szintenként adhatjuk meg, hogy mely tagok jelenjenek meg az osztály ábrázolásán, VS-ban a tagokat külön-külön elrejthetjük.

A ***láthatóságot*** az EA a szabványos módon (8. ábra), a VS külön jelekkel ábrázolja. A ***statikus tagokat*** aláhúzással jelzi az UML. ***Kezdőérék*** megadása az = jel után történhet.

A ***változó típusának*** megadása a *változónév : típus* szintaktikával történik. Figyelem! Ez elért a C nyelvekben megszokottól. A diagram részletességétől függ, hogy megadjuk-e a mezők típusát és a metódusok paramétereit, visszatérési értékét (és ezek típusait). (8. ábra)




```
public class MyClass
{
    private const string privateField="This is a private constant.";
    protected string protectedMethod(string parameter);
    internal static string InternalMethod();
    public string PublicProperty { get; set; }
}
```

8. ábra Osztályok tagjainak jelölése

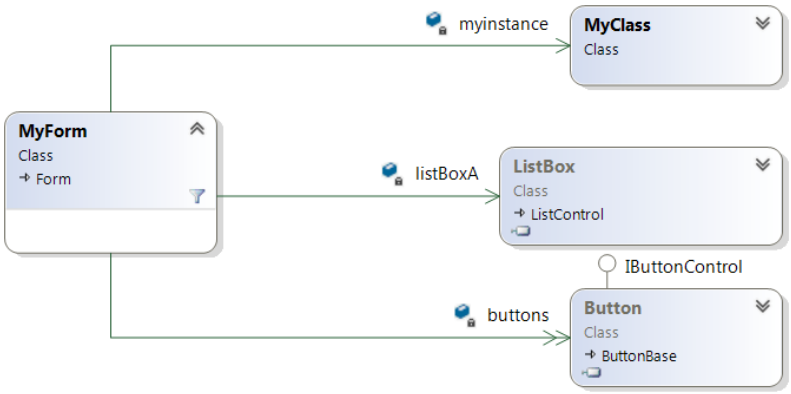
Osztályok közötti kapcsolatok

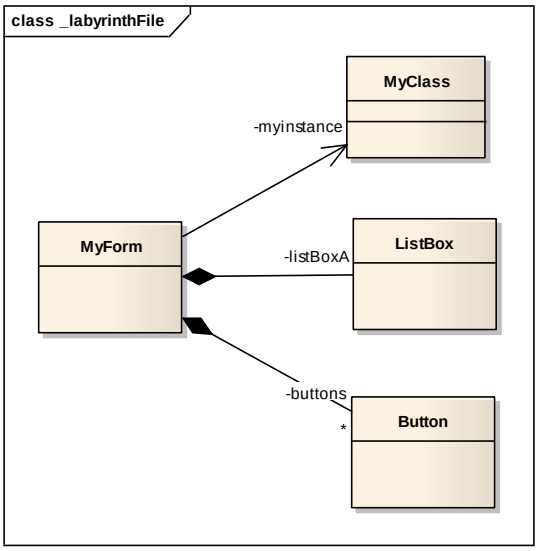
Generalizáció: két osztály közötti általános-speciális viszonyt fejez ki. Az objektumorientált nyelvek öröklődési kapcsolatát jelzi. Jele: üresfejű háromszög. A 2. ábra Button és ButtonBase osztálya között látható generalizációs kapcsolat (ős/általános: ButtonBase, utód/speciális: Button).

Az asszociációs, az aggregációs és a kompozíciós kapcsolat a kódban adattagként jelenik meg. Az ismerő/tartalmazó osztály adattagja a másik (ismert/tartalmazott) osztály egy példánya. Tehát a mezőket ábrázolhatjuk asszociációként is. A kapcsolaton megadhatjuk a mező nevét (szerepnév) és annak láthatóságát. Mivel a kódban nincs különbség a három kapcsolat között, a VS nem ábrázolja a különbséget. Alapos tervezésnél azonban szükség van a kapcsolatok megkülönböztetésére (pl. a kompozíciós kapcsolatnál a tartalmazó felel a tartalmazott példány felszabadításáért, ezért ilyen esetekben a szabványos UML jelöléseket kell használni tervezőszoftverben.

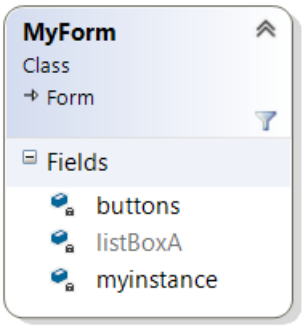
A kapcsolatban részt vevő ***példányok számát (multiplicitását)*** az UML-ben egyértelműen (számmal) megadhatjuk, VS-ban csak tudjuk jelölni, hogy egy vagy több objektum kapcsolódik az osztályhoz.

A 9. ábra három diagramja ugyanazt a kódot jeleníti meg: a formnak tagja egy ListBox, egy MyClass példány és egy Button lista. EA-ben tudjuk megjeleníteni a kompozíciót, vagyis, hogy a Form szerves részei a vezérlők, a Formnak gondoskodnia kell megszűnésekor a vezérlők nem felügyelt részének felszabadításáról (Dispose pattern).



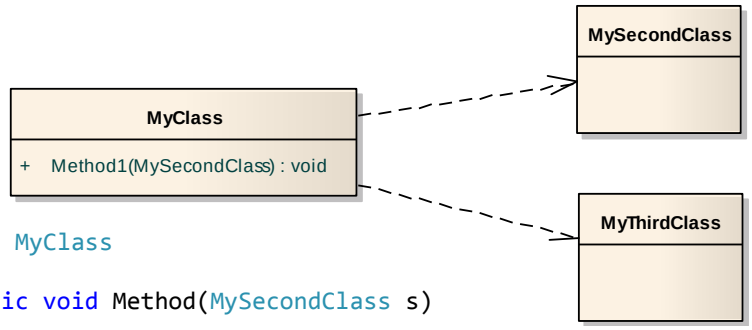



```
public partial class MyForm : Form
    {
        List<Button> buttons;
        MyClass myinstance;
        ListBox listBoxA;
    }
```



9. ábra Asszociáció, aggregáció és kompozíció jelölése

Függőség van két osztály között, ha az egyik használja (metódusának paramétere vagy visszatérési értéke, esetleg metódusában változó típusa) a másikat. Ezt UML-ben szaggatott nyíllal jelöljük. A VS nem jelöli ezt a fajta kapcsolatot (minden függőségi viszonyt így jelöl az UML). (10. ábra)




```
public class MyClass
    {
        public void Method(MySecondClass s)
        {
            MyThirdClass th;
                    …
        }
    }
```

10. ábra Függőség jelölése