

Szoftverttechnikák önálló gyakorlat

2. önálló feladat

Modern nyelvi eszközök

Modern nyelvi eszközök gyakorlása:
tulajdonság, delegate, esemény, attribútum,
generikus osztályok

A gyakorlatot kidolgozta: Kővári Bence, Benedek Zoltán

Utolsó módosítás ideje: 2020.03.01, Benedek Zoltán

Tartalom

TARTALOM	2
BEVEZETÉS	2
FELADAT 1 – BALJÓS ÁRNYAK	4
FELADAT 2 – A KLÓNOK TÁMADÁSA	4
FELADAT 3 – A SITH-EK BOSSZÚJA.....	5
FELADAT 4 – DELEGÁTOK	8
EMLÉKEZTETŐK.....	9
PLUSZ FELADAT – 2 IMSC PONTÉRT.....	9
TOVÁBBIK.....	10

Bevezetés



Az önálló feladat a 2. előadáson elhangzottakra épít. A feladatok elméleti háttéréül az „Előadás 02 - Modern nyelvi eszközök.pdf” leírás, gyakorlati háttéréül a „2. gyakorlat - Modern nyelvi eszközök” laborgyakorlat szolgál.

A fentiekre építve, jelen önálló gyakorlat feladatai a feladatlírást követő rövidebb iránymutatás segítségével elvégezhetők.

Az önálló gyakorlat célja:

- Tulajdonságok és események használatának gyakorlása
- Az attribútumok használatának gyakorlása
- Alapvető gyűjteménytípusok használatának gyakorlása

A feladat publikálásának és beadásának alapelvei többségében megegyeznek az 1. házi feladatával, a részletek annak útmutatójában találhatók. Pár új, illetve változott követelmény:

- A feladathoz tartozó GitHub Classroom hivatkozás: <https://classroom.github.com/a/fURYH-ue>
A munkamenet megegyezik az első házi feladatével: a fenti hivatkozással mindenkinek születik egy privát repója, abban kell dolgozni és a határidőig a feladatot beadni.
-  **Lényeges**, hogy az elindulással ne várd meg a határidő közeledtét, legalább a saját repó létrehozásáig juss el előbb. Így ha bármi elakadás lenne, még időben tudunk segíteni.
-  **Változás!** A repository gyökérmappájában található neptun.txt fájlba írd bele a Neptun kódod, csupa nagybetűvel. **De csak akkor, ha a megoldással elkészültél, a feladatot beadottnak tekinted.**

- **i Változás!** A beadott megoldások mellé külön indoklást, illetve leírást nem várunk el, ugyanakkor kérjük, hogy a beadott kódban a témakörhöz kapcsolódó részek (property, attribute, event, delegate) **kommentekkel legyenek ellátva**.
- A megoldás során törekedni kell az egységbezárásra (a publikus tagváltozók alkalmazása nem elfogadható).
- A munka során a kiindulási repóban levő solutionben/projektben kell dolgozni, új projektet/solutiont ne hozz létre.
- Amikor a házi feladatod beadottnak tekinted, célszerű ellenőrizni a GitHub webes felületén a repository-ban a fájlokra való rápillantással, hogy valóban minden változtatást push-oltál-e.
- Emlékeztető: *valamennyi házi feladatot a megadott határidőig a tanszéki portálra (<https://www.aut.bme.hu/>) is fel kell tölteni egy zip csomagban!*
 - A tanszéki portálra feltöltendő zip állománynak tartalmaznia kell a megoldás(ok) forráskódját. További szabályok:
 - A zip állomány nem tartalmazhatja a kimeneti („.exe”) és köztes állományokat! Ezen állományok törléséhez a Solution mappából kézzel törölni kell a „bin” és „obj” mappákat teljes tartalmukkal együtt.
 - A zip állományba ne tegyük bele a „.git” és „.vs” könyvtárat.

Feladat 1 – Baljós árnyak

Feladat

Amint az közismert, a jedi lovagok erejét a sejtjeikben élő kis életformák, a midi-chlorianok adják. Az eddigi legmagasabb midi-chlorian szintet (20.000 fölötti értéket) Anakin Skywalkerénél mérték.

Készítsen egy osztályt `Jedi` néven mely egy `string` típusú `Name` és egy `integer` típusú `MidiChlorianCount` tulajdonsággal rendelkezik. Utóbbi esetében figyeljen rá, hogy a `MidiChlorianCount` értékét ne lehessen **10**-re, vagy annál kisebb értékre állítani, ha ezzel próbálkozik valaki, az osztálynak kivételt kell dobnia. A validáció során a lehető legegyszerűbb, legletisztultabb megoldást válaszod: a property setterben egyszerű if-et használj és dobj kivételt, ne legyen az if-nek else ága, valamint nincs szükség a return használatára sem.

Megoldás

A feladat megoldása a 2. gyakorlat 1. feladatával analóg módon készíthető el. A `MidiChlorianCount` tulajdonság setterében érvénytelen érték esetén dobjunk kivételt. Ezt például a következő utasítással tehetjük meg:

```
throw new ArgumentException("You are not a true jedi!");
```

Feladat 2 – A klónok támadása

Feladat

Egészítse ki az 1. feladatban elkészített osztályt attribútumokkal úgy, hogy amennyiben az `XmlSerializer` osztály segítségével, XML formátumú adatfájlt állítunk elő belőle, a tulajdonságai egy-egy XML attribútum formájában, magyarul jelenjenek meg! Ezt követően írjon egy függvényt, mely a `Jedi` osztály egy példányát egy szövegfájlba sorosítja, majd onnan visszaolvassa egy új objektumba (ezzel tulajdonképpen klónozza az eredeti objektumot).

i Fontos: a mentést és betöltést végző/demonstráló kódot írd egy közös, erre dedikált függvénybe, a függvényt pedig lásd el a `[Description("Feladat2")]` C# attribútummal (a függvény előtti sorba kell beírni). A mentett/betöltött objektum lokális változóként legyen ebben a függvényben megvalósítva. Az osztály/függvény neve bármi lehet (pl. kerülhet a `Program` osztályba is). A függvény nem szorosan a feladathoz tartozó kódot ne tartalmazzon, így más (rész)feladathoz tartozót sem. A függvényt hívd meg a `Program` osztály `Main` függvényéből.

Megjegyzés: a fenti attribútum használatához `using-olni` kell a `System.ComponentModel` névteret.

Tipp: Az Xml sorosítást szabályozó attribútumokat ne tagváltozók, hanem a property-k felett helyezd el!

i Fontos: A `[Description("Feladat2")]` attribútum csak egyetlen függvény fölött szerepelhet!

Megoldás

A feladat megoldása a 2. gyakorlat 4. feladatával analóg módon készíthető el. A megoldáshoz az alábbi segítségeket adjuk

- A sorosítást követően az XML fájlnek ehhez hasonlóan kell kinéznie:

```
<?xml version="1.0"?>
<Jedi xmlns:xsi="..." Nev="Obi-Wan" MidiChlorianSzam="15000" />
```

Lényeges, hogy az egyes Jedik „Jedi” XML elemként, nevük „Nev”, a midichlorianszámuk „MidiChlorianSzam” XML attribútumként jelenjen meg.

- A sorosított objektumok visszatöltésére a labor során nem néztünk példakódot, ezért ezt itt megadjuk:

```
XmlSerializer ser = new XmlSerializer(typeof(Jedi));
FileStream fs = new FileStream("jedi.txt", FileMode.Open);
Jedi clone = (Jedi)ser.Deserialize(fs);
fs.Close();
```

Az előző művelet sor először létrehoz egy sorosítót (`ser`), mellyel majd a beolvasást később elvégezzük. A beolvasást egy „jedi.txt” nevű fájlból fogjuk végezni, amelyet a második sorban olvasásra nyitunk meg (figyeljük meg, hogy ha írni akartuk volna, akkor `FileMode.Create`-et kellett volna megadni).

Feladat 3 – A Sith-ek bosszúja

Feladat

A Jeditanácsban az utóbbi időben nagy a fluktuáció. Hogy a változásokat könnyebben nyomon követhessük, készítsen egy osztályt, mely képes nyilvántartani a tanács tagjait és minden változásról egy esemény formájában szöveges értesítést küldeni! A lista manipulációját két függvénnyel lehessen végezni. Az `Add` függvény egy új jedi lovagot regisztráljon a tanácsba, míg a `Remove` függvény távolítsa el a **legutoljára** felvett tanácstagot. Külön értesítés jelezze, ha a tanács teljesen kiürül.

A tanács tagok (`members`) nyilvántartását egy `List<Jedi>` típusú tagváltozóban tároljuk, az `Add` függvény ehhez a listához fűzze hozzá az új elemeket, míg a `Remove` függvény generikus lista `RemoveAt` utasításával mindig a **legutoljára** felvett tagot távolítsa el (az utolsó elem indexét a lista hossza alapján tudjuk meghatározni, amit a `Count` property ad vissza).

Az értesítés egy C# eseményen (C# event) keresztül történjen. Az eseményhez tartozó `delegate` típus paraméterként egy egyszerű `string`-et kapjon. Az új tag hozzáadását, az egyes tagok eltávolítását, illetve az utolsó tag eltávolítását más-más szövegű üzenet jelezze.

Az esemény típusának ne használjon beépített `delegate` típust, hanem vezessen be egy sajátot. Az esemény elsütését közvetlenül az `Add` és a `Remove` műveletekben végezze (ne vezessen be erre segédfüggvényt).

i Fontos: A *Jeditanács* objektumot létrehozó és azt tesztelő (C# eseményére való feliratkozás, *Add* és *Remove* hívása) kód kerüljön egy közös, önálló függvénybe, ezt a függvényt pedig lásd el a `[Description("Feladat3")]` C# attribútummal. Az osztály/függvény neve bármi lehet. A függvény nem szorosan a feladathoz tartozó kódot ne tartalmazzon, így más (rész)feladathoz tartozót sem. A függvényt hívd meg a *Program* osztály *Main* függvényéből.

Megjegyzés: A `[Description("Feladat3")]` attribútum csak egyetlen függvény fölött szerepelhet.

Megoldás

A feladat megoldása a 2. gyakorlat több részletére is épít. Az új esemény bevezetését a 2. és a 3. feladatban leírt módon tudjuk elvégezni, míg a tanács tagjait egy listában tudjuk nyilvántartani. A fenti információk alapján próbáld meg **önállóan** megoldani a feladatot, majd ha készen vagy, a következő oldalon folytasd az útmutató olvasását és vedd össze a megoldásodat a lenti referencia megoldással! Szükség szerint korrigáld a saját megoldásod!

Tipp: a példa épít arra, hogy a résztvevő osztályok, tulajdonságok, delegate-ek publikus láthatóságúak. Amennyiben fura fordítási hibával találkozol, vagy az *XmlSerializer* futásidőben hibát dob, első körben azt ellenőrizd, hogy minden érintett helyen megfelelően beállítottad-e a publikus láthatóságot.

A referencia megoldás lépései a következők:

1. Hozzunk létre egy új osztályt, `JediCouncil` néven
2. Vegyünk fel egy `List<Jedi>` típusú mezőt és inicializáljuk egy üres listával
3. Valósítsuk meg az *Add* és a *Remove* függvényeket

A fenti lépéseket követően az alábbi kódot kapjuk:

```
class JediCouncil
{
    List<Jedi> members = new List<Jedi>();
    public void Add(Jedi newJedi)
    {
        members.Add(newJedi);
    }
    public void Remove()
    {
        // Eltávolítja a lista utolsó elemét
        members.RemoveAt(members.Count-1);
    }
}
```

Következő lépésként valósítsuk meg az eseménykezelést.

4. Definiáljunk egy új delegát típust, mely az értesítések szövegét adja majd át:

```
public delegate void CouncilChangedDelegate(string message);
```

5. Egészítsük ki a `JediCouncil` osztályt az eseménykezelővel:

```
public class JediCouncil
{
    public event CouncilChangedDelegate CouncilChanged;
    ...
}
```

6. Süssük el az eseményt, amikor új tanácstagot veszünk fel. Ehhez az `Add` metódust kell kiegészítenünk.

```
public void Add(Jedi newJedi)
{
    members.Add(newJedi);
    if (CouncilChanged != null)
        CouncilChanged("Új taggal bővültünk");
}
```

7. Süssük el az eseményt, amikor egy tanácstag távozik! Különböztessük meg azt az esetet, amikor a tanács teljesen kiürül. Ehhez a `Remove` metódust kell kiegészítenünk.

```
public void Remove()
{
    // Eltávolítja a lista utolsó elemét
    members.RemoveAt(members.Count-1);
    if (CouncilChanged!=null)
    {
        if (members.Count > 0)
            CouncilChanged("Zavart érzek az erőben");
        else
            CouncilChanged("A tanács elesett!");
    }
}
```

8. Megoldásunk teszteléséhez vegyünk fel egy `MessageReceived` függvényt abba az osztályba, ahol az eseményre való feliratkozást és az esemény kezelését tesztelni szeretnénk (pl. a `Program` osztályba). Ezt a függvényt fogjuk feliratkoztatni a `JedCouncil` értesítéseire.

```
class Program
{
    static void MessageReceived(string message)
    {
        Console.WriteLine(message);
    }
}
```

9. Végezetül teszteljük az új osztályunkat egy erre a célra dedikált függvény megírásával (ez történhet pl. a Program osztályban), a függvény fölé tegyük oda a **[Description("Feladat3")]** attribútumot! A függvény váza:

```
// Tanács létrehozása
JediCouncil council = new JediCouncil();
council.CouncilChanged += MessageReceived;

<Itt adj hozzá két Jedi objektumot a council objektumhoz az Add hívásával>
...


council.Remove();
council.Remove();
```

10. Ha jól végeztük a dolgunkat, a program futtatását követően a következő kimenetet kell kapnunk:


```
Új taggal bővültünk
Új taggal bővültünk
Zavart érzek az erőben
A tanács elesett!
```

Feladat 4 – Delegátok

Feladat

Egészítsük ki a JediCouncil osztályt egy `GetBeginners()` paraméter nélküli függvénnyel, mely visszatérési értékében visszaadja a Jedi tanács összes olyan tagját, melynek a midi-chlorian száma 300 alatt van! A függvényen belül a tagok kikeresésére használjuk a `List<Jedi>` osztály `FindAll()` függvényét. Írjunk egy dedikált függvényt is (pl. a Program osztályba), mely meghívja a `GetBeginners()` függvényünket és kiírja a visszaadott jedi lovagok neveit! Ez a függvény nem szorosan a feladathoz tartozó kódot ne tartalmazzon, így más (rész)feladathoz tartozót sem.  Ezt a „tesztelő” függvényt lásd el a **[Description("Feladat4")]** C# attribútummal. A függvényt hívd meg a Program osztály Main függvényéből.

Megjegyzés: A **[Description("Feladat4")]** attribútum csak egyetlen függvény fölött szerepelhet.

 A megvalósítás során vezess be egy külön statikus metódust (pl. a Program osztályba), mely paraméterként egy Jeditanács objektumot kap, abba legalább három felparaméterezett Jedi objektumot az Add hívásával felvesz. A célunk ezzel az, hogy egy olyan inicializáló metódusunk legyen, mely a későbbi feladat(ok) során is felhasználható, ne kelljen a kapcsolódó inicializáló kódot duplikálni.

Megoldás

A feladat megoldásához a 2. gyakorlat 6. feladatát használhatjuk referenciaként. Segítségként megadjuk a következőket

- A függvényünk akár több találatot is visszaadhat, ezért a visszatérési érték típusa `List<Jedi>`
- A `FindAll` paraméterként az esetünkben egy `bool` Függvénynév(`Jedi`) szignatúrájú szűrőfüggvényt vár el.

Emlékeztetők


Mielőtt véglegesen beadnád a házi feladatot, mindenképpen ellenőrizd a következőket:

- A beadott megoldások mellé külön indoklást, illetve leírást nem várunk el, ugyanakkor kérjük, hogy a beadott kódban a témakörhöz kapcsolódó (property/delegate/event) részek **kommentekkel legyenek ellátva**.
- A megfelelő függvényeket lásd el "Feladat2", "Feladat3", "Feladat4" Description attribútummal. Ellenőrizd vissza, egy se maradt le, esetleges copy-paste-után átnevezted-e, stb. **Mindegyik csak egyetlen függvény fölött szerepelhet**, és lehetőleg egyik se legyen a `Jedi` és `JediTanács`ot reprezentáló osztályokban.
- Publikus tagváltozókat ne használj.

Plusz feladat – 2 iMsc pontért

Feladat

Bővítsük ki a `JediCouncil` osztályt.

- Készíts egy `Count` nevű integer visszatérési értékű property-t (tulajdonságot), amely minden lekérdezéskor a tanácsban aktuálisan található Jedi-k számát adja vissza. Ügyelj arra, hogy ezt az értéket csak lekérdezni lehessen (beállítani nem).
 - Tipp: a `JediCouncil`-ban található `members` nevű tagváltozónak van egy `Count` nevű property-je, a megoldás építsen erre.
- Készíts egy `CountIf` nevű függvényt, amely szintén a tanácsstagok megszámolására való, de csak bizonyos feltételnek eleget tevő tanácsstagokat vesz figyelembe. A függvény visszatérési értéke integer, és a feltételt, amelynek megfelelő tanácsstagok számát visszaadja, egy delegate segítségével kap meg paraméterként (tehát a `CountIf`-nek kell legyen paramétere).
- A property és a függvény működését demonstráld egy erre dedikált közös függvényben,  amit láss el a **[Description("Feladat5")]** attribútummal. Ez a függvény nem szorosan a feladathoz tartozó kódot ne tartalmazzon, viszont a JediTanács feltöltéséhez az előző feladatban bevezetett segédfüggvényt hívd. A függvényt hívd meg a Program osztály Main függvényéből.
Megjegyzés: A **[Description("Feladat5")]** attribútum csak egyetlen függvény fölött szerepelhet.

Megoldás

- A Count nevű property esetében csak a get ágak van értelme, ezért a set ágakat meg se írjuk. Ez egy csak olvasható tulajdonság legyen.
- A CountIf függvény megírásában a 4-es feladat nyújt segítséget. A különbség, hogy a CountIf nem a tanácstagokat, csak a darabszámot adja vissza.
 - Tipp: a CountIf függvény a feltételt paraméterként egy bool Függvénynév(Jedi) szignatúrájú szűrőfüggvényt várjon.

Továbbiak

Pár jótanács, ismétlésképpen (a többségük a későbbi házi feladatokra is vonatkozik):

- A repository gyökérmappájában található neptun.txt fájlba írd bele a Neptun kódod, csupa nagybetűvel. A fájlban csak ez a hat karakter legyen, semmi más.
- A GitHub-ról leöltött kiinduló solutionben/projektekben kell dolgozni, nem újonnan létrehozottban.
- A megoldásod lásd el kódkommentekkel.
- Lényeges, hogy a feladatok csak akkor kerülnek elfogadásra, ha teljesen elkészülnek, és minden tekintetben teljesítik a követelményeket. Nem forduló kód, illetve részleges megoldás elfogadásában nem érdemes bízni.