2. HF - Nyelvi eszközök

Bevezetés

Az önálló feladat a 2. előadáson és a 3. előadás első felében elhangzottakra épít (ezek a "Előadás 02 - Nyelvi eszközök" előadásanyagban szerepelnek). Gyakorlati hátteréül a 2. labor - Nyelvi eszközök laborgyakorlat szolgál.

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

Az önálló gyakorlat célja:

- Tulajdonságok (property) használatának gyakorlása
- Delegátok (delegate) és események (event) alkalmazása
- .NET attribútumok használatának gyakorlása
- Alapvető gyűjteménytípusok használatának gyakorlása
- Lambda kifejezések gyakorlása

A szükséges fejlesztőkörnyezetről itt található leírás.

> **C# 12-es (és újabb) nyelvi elemek használata**
>
> A házi feladat megoldása során C# 12-es, és annál újabb nyelvi elemek, (pl. primary constructor) nem használhatók, ugyanis a GitHub-on futó ellenőrző ezeket még nem támogatja.

Beadás menete, előellenőrző

A beadás menete megegyezik az első házi feladatéval (részletes leírás a szokásos helyen, lásd Házi feladat munkafolyamat és a Git/GitHub használata):

1. GitHub Classroom segítségével hozz létre magadnak egy repository-t. A meghívó URL-t Moodle-ben találod (a tárgy nyitóoldalán a "*GitHub classroom hivatkozások a házi feladatokhoz*" hivatkozásra kattintva megjelenő oldalon látható). Fontos, hogy a megfelelő, ezen házi feladathoz tartozó meghívó URL-t használd (minden házi feladathoz más URL

tartozik).

2. Klónozd le az így elkészült repository-t. Ez tartalmazni fogja a megoldás elvárt szerkezetét.
3. A feladatok elkészítése után commit-old és push-old a megoldásod.

Az előellenőrző is a szokásos módon működik. Részletes leírás: A házi feladat előellenőrzése és hivatalos értékelése.

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 Skywalkernél mérték.

Készíts egy osztályt `Jedi` néven mely egy `string` típusú `Name` és egy `int` típusú `MidiChlorianCount` tulajdonsággal rendelkezik. Utóbbi esetében figyelj rá, hogy a `MidiChlorianCount` értékét ne lehessen 35-re, vagy annál kisebb értékre állítani, ha ezzel próbálkozik valaki, az osztálynak kivételt kell dobnia. A validáció során a lehető legegyszerűbb, legletisztultabb megoldást válaszd: 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. labor 1. feladatával analóg módon készíthető el. A `MidiChlorianCount` tulajdonság setterében érvénytelen érték esetén dobj kivételt. Ezt például a következő utasítással tehető meg:

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

Feladat 2 – A klónok támadása

Feladat

Egészítsd ki az 1. feladatban elkészített osztályt attribútumokkal úgy, hogy amennyiben az `XmlSerializer` osztály segítségével, XML formátumú adatfájlba írunk/sorosítunk ki egy `Jedi` objektumot, a tulajdonságai egy-egy XML attribútum formájában, magyarul jelenjenek meg! Ezt követően írj egy függvényt, mely a `Jedi` osztály egy példányát egy szövegfájlba sorosítja, majd onnan visszaolvassa egy új objektumba (ezzel tulajdonképpen klónozva az eredeti objektumot).

XML sorosító attribútumai

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

A Jedi osztály legyen publikus

Az XML sorosító csak publikus osztályokon tud dolgozni, ennek megfelelően a Jedi osztály legyen publikus:

```
public class Jedi { ...}
```

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

Lényeges, hogy

- az attribútumot függvény, és NE osztály fölé írd,
- az attribútumot ne a logikát megvalósító, hanem a tesztelést végző függvény fölé írd,
- **az attribútum csak egyetlen függvény fölött szerepelhet.**

Megoldás

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

- A sorosítást követően az XML fájlnak 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:

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

Az előző műveletsor először létrehoz egy sorosítót (`serializer`), 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, készíts 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 (ehhez ugyanazt az eseményt használd, mint a többi változás esetén, csak más szöveggel jelezze).

A tanácstagok (`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, melyet 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 elsütését közvetlenül az `Add` és a `Remove` műveletekben végezd el (ne vezess be erre segédfüggvényt).

Az esemény típusának ne használj beépített delegate típust, hanem vezess be egy sajátot.

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.

Lényeges, hogy

- az attribútumot függvény, és NE osztály fölé írd,
- az attribútumot ne a logikát megvalósító, hanem a tesztelést végző függvény fölé írd,
- az attribútum csak egyetlen függvény fölött szerepelhet.

Megoldás

A feladat megoldása a 2. labor 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 kinyitható blokkban folytasd az útmutató olvasását és vesd össze a megoldásodat a lenti referencia megoldással! Szükség szerint korrigáld a saját megoldásod!

Publikus láthatóság

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.

Referencia megoldás

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:

```
public 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 (az osztályon kívül, mivel ez is egy típus), 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);
```

```
    // TODO: Itt süsd el az eseményt.
    // Figyelj arra, hogy csak akkor tedd meg, ha van legalább egy feliratkozó/
előfizető.
    // Ennek során ne a terjengősebb null ellenőrzést, hanem a modernebb,
?.Invoke-ot használd.
}
```

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);

    // TODO: Itt süsd el az eseményt.
    // Figyelj arra, hogy csak akkor tedd meg, ha van legalább egy feliratkozó/
előfizető.
}
```

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 `JediCouncil` értesítéseire.

Program.cs

```
private 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
var council = new JediCouncil();

// TODO: Itt iratkozz fel a council CouncilChanged eseményére

// TODO 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!
```

Események null vizsgálata

Amennyiben a `JediCouncil.Add` műveletben `null` vizsgálattal végezted annak ellenőrzését, hogy van-e legalább egy feliratkozó az eseményre, ezt alakítsd át korszerűbb megoldásra (`?.Invoke` alkalmazása, mely tömörebb formában szintén elvégzi az ellenőrzést, de `null` vizsgálat nélkül – erről a kapcsolódó előadáson és laboron is volt szó). Ezt elég a `JediCouncil.Add` kapcsán megtenni, a `JediCouncil.Remove` esetében mindkét megoldás elfogadható most.

Feladat 4 – Delegátok

Feladat

Egészítsd ki a `JediCouncil` osztályt egy olyan paraméter nélküli függvénnyel (**a függvénynév végződjön `_Delegate`-re, ez kötelező**), mely visszatérési értékében visszaadja a Jedi tanács összes olyan tagját, melynek a midi-chlorian száma **530** alatt van!

- Függvényt használj, ne tulajdonságot a lekérdezésre.
- A függvényen belül a tagok kikeresésére használd a `List<Jedi>` osztály `FindAll()` függvényét.
- Ebben a feladatban még **NEM használhatsz lambda kifejezést**!

Írj egy dedikált „tesztelő” függvényt is (pl. a `Program` osztályba), mely meghívja a fenti 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.

Fontos

Ezt a „tesztelő” függvényt lásd el a `[Description("Feladat4")]` C# attribútummal. A függvényt hívd meg a `Program` osztály `Main` függvényéből.

Lényeges, hogy

- az attribútumot függvény, és NE osztály fölé írd,
- az attribútumot ne a logikát megvalósító, hanem a tesztelést végző függvény fölé írd,
- az attribútum csak egyetlen függvény fölött szerepelhet.

Inicializáció kiszervezése

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. labor 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 j)` szignatúrájú szűrőfüggvényt vár el.

Feladat 5 – Lambda kifejezések

A feladat megfelel az előzőnek, csak most lambda kifejezés segítségével fogunk dolgozni. Ez a témakör szerepelt előadáson és laboron is (2. labor 6. feladat).

Egészítsd ki a JediCouncil osztályt egy olyan paraméter nélküli függvénnyel (**a függvénynév végződjön `_Lambda`-ra, ez kötelező**), mely visszatérési értékében visszaadja a Jedi tanács összes olyan tagját, melynek a midi-chlorian száma 1000 alatt van!

- Függvényt használj, ne tulajdonságot a lekérdezésre.

- A függvényen belül a tagok kikeresésére használd a `List<Jedi>` osztály `FindAll()` függvényét.
- Ebben a feladatban kötelezően lambda kifejezést kell használj (az mindegy, hogy statement vagy expression lambdát)!

Írj egy dedikált „tesztelő” függvényt is (pl. a `Program` osztályba), mely meghívja a fenti 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.

Fontos

Ezt a „tesztelő” függvényt lásd el a `[Description("Feladat5")]` C# attribútummal. A függvényt hívd meg a `Program` osztály `Main` függvényéből.

Lényeges, hogy

- az attribútumot függvény, és NE osztály fölé írd,
- az attribútumot ne a logikát megvalósító, hanem a tesztelést végző függvény fölé írd,
- az attribútum csak egyetlen függvény fölött szerepelhet.

Feladat 6 – `Action` / `Func` használata

Ez a feladat a 3. előadás anyagára épít, laboron (idő hiányában) nem szerepelt. Ettől függetlenül ez egy lényeges alaptémakör a tárgyban.

A projektbe vegyél fel egy `Person` és egy `ReportPrinter` osztályt (egy-egy, az osztály nevével egyező fájlba, az alapértelmezett, `ModernLangToolsApp` névtérbe), a következő tartalommal:

Person és ReportPrinter osztályok

```
class Person
{
    public Person(string name, int age)
    {
        Name = name;
        Age = age;
    }

    public string Name { get; set; }
    public int Age { get; set; }
}
```

```
class ReportPrinter
{
    private readonly IEnumerable<Person> people;
    private readonly Action headerPrinter;

    public ReportPrinter(IEnumerable<Person> people, Action headerPrinter)
    {
        this.people = people;
        this.headerPrinter = headerPrinter;
    }

    public void PrintReport()
    {
        headerPrinter();
        Console.WriteLine("----------------------------------------");
        int i = 0;
        foreach (var person in people)
        {
            Console.Write($"{++i}. ");
            Console.WriteLine("Person");
        }
        Console.WriteLine("--------------- Summary -----------------");
        Console.WriteLine("Footer");
    }
}
```

Ez a `ReportPrinter` osztály arra használható, hogy a konstruktorában megadott személyek adatairól formázott riportot írjon ki a konzolra fejléc/adatok/lábléc hármas bontásban. A `Program.cs` fájlba vedd fel az alábbi függvényt a `ReportPrinter` kipróbálására, és ezt hívd is meg a `Main` függvényből:

ReportPrinter tesztelése

```
[Description("Feladat6")]
static void test6()
{
    var employees = new Person[] { new Person("Joe", 20), new Person("Jill", 30) };

    ReportPrinter reportPrinter = new ReportPrinter(
        employees,
        () => Console.WriteLine("Employees")
        );

    reportPrinter.PrintReport();
}
```

Futtassuk az alkalmazást. Az alábbi kimenetet kapjuk a konzolon:

```
Employees
----------------------------------------
1. Person
2. Person
--------------- Summary -----------------
Footer
```

Az első sorban "----" felett található a fejléc. Alatta az egye személyekhez egy-egy "Person" beégetett szöveg, majd a "----" alatt a lábléc, egyelőre csak egy beégetett "Footer" szöveggel.

A megoldásban látható, hogy a fejléc szövege a `ReportPrinter` osztályba nincs beégetve. Ezt `ReportPrinter` felhasználója adja meg konstruktor paraméterben egy delegate, esetünkben egy lambda kifejezés formájában. A delegate típusa a .NET beépített `Action` típusa.

A feladatok a következők:

Warning

A megoldás során NEM használhatsz saját delegate típust (a .NET beépített delegate típusaival dolgozz, a megoldás csak ekkor elfogadható).

1. Alakítsd át a `ReportPrinter` osztályt úgy, hogy az osztály használója ne csak a fejlécet, hanem a láblécet is meg tudja adni egy delegate formájában a konstruktorban.
2. Alakítsd tovább a `ReportPrinter` osztályt úgy, hogy az egyes személyek kiírásakor ne a fix "Person" szöveg jelenjen meg, hanem a `ReportPrinter` osztály használója tudja az egyes

személyek adatait az igényeinek megfelelően kiírni a konzolra egy konstruktorban megadott delegate segítségével (a fix "Person" helyett). Lényeges, hogy a sorszám a sor elején mindig meg kell jelenjen, ez nem lehet a `ReportPrinter` használója által megváltoztatható (vagyis ezt a továbbiakban is a `ReportPrinter` osztálynak kell kiírnia)!

> **Tipp a megoldáshoz**
>
> Hasonló megközelítésben gondolkozz, mint a fejléc és lábléc esetében, de itt ehhez a `ReportPrinter` felhasználójának meg kell kapnia a személy objektumot ahhoz, hogy azt formázottan ki tudja írni a konzolra.

3. A `Program.cs` fájlban a `ReportPrinter` használatát alakítsd úgy (megfelelő lambda kifejezések megadásával), hogy a kimenet a konzolon a következő legyen:

```
Employees
-----------------------------------------
1. Name: Joe (Age: 20)
2. Name: Jill (Age: 30)
--------------- Summary -----------------
Number of Employees: 2
```

> **Láblécben a dolgozók számának kiírása**
>
> Ahhoz, hogy a láblécben a dolgozók számának kiírását elegáns módon meg tudd tenni, szükség van a "variable capturing" témakör ismeretére (lásd 3. előadás "Variable capturing, closure" fejezet).

> **Házi feladat ellenőrzése**
>
> A "Feladat 6" feladatot, vagyis azt, hogy a `ReportPrinter`-t és annak használatát jól alakítottad-e át, a GitHub-os automata ellenőrző NEM ellenőrzi. Teszteld a megoldásod alaposan, hogy ne csak a határidő után utólag, a házi feladatok manuális ellenőrzése során derüljön ki, hogy a megoldás nem elfogadható. (Kiegészítés: 2024.03.13 reggeltől kezdve már erre is van részleges automata ellenőrzés)

4. A következő feladat opcionális, a beépített `Func` delegate-ek gyakorlására ad jó lehetőséget. A `ReportPrinter` osztálynak van egy komolyabb hátránya: a kimeneti riportot csak a konzolon tudjuk a segítségével megjeleníteni. Rugalmasabb megoldás lenne, ha nem írna a konzolra, hanem egy string formájában lehetne a segítségével a riportot előállítani. Ezt a stringet már úgy használhatnánk fel, ahogy csak szeretnénk (pl. írhatnánk fájlba is).

A feladat a következő: vezess be egy `ReportBuilder` osztályt a már meglévő `ReportPrinter` mintájára, de ez ne a konzolra írjon, hanem egy a teljes riportot tartalmazó stringet állítson elő, melyet egy újonnan bevezetett, `GetResult()` művelettel lehessen tőle lekérdezni.

> **Beadás**
>
> Ha beadod a feladatot, a `ReportBuilder`-t példányosító/tesztelő kódot ne a fenti, `test6` függvénybe tedd, hanem vezess be egy `test6b` nevű függvényt, és lásd el a `[Description("Feladat6b")]` attribútummal.

> **Tippek a megoldáshoz**
>
> - Célszerű az osztályba egy `StringBuilder` tagváltozót bevezetni, és ennek segítségével dolgozni. Ez nagyságrenddel hatékonyabb, mint a stringek "+"-szal való összefűzögetése.
> - A `ReportBuilder` osztály használója itt már ne a konzolra írjon, hanem megfelelő beépített típusú delegate-ek (itt az `Action` nem lesz megfelelő) segítségével adja vissza a `ReportBuilder` számára azokat a stringeket, melyeket bele kell fűznie a kimenetbe. A tesztelés során most is lambda kifejezéseket használj!

Feladat 7 (IMSc) – beépített `Func` / `Action` generikus delegate típusok használata

A feladat megoldása nem kötelező, de erősen ajánlott: alapanyag, így ZH-n/vizsgán szerepelhet. Laboron nem volt, csak előadáson.

A megoldásért +2 IMSc pont is jár.

Feladat

Bővítsd ki a `JediCouncil` osztályt.

- Készíts egy `Count` nevű `int` 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 tanácstagok megszámlálására való, de csak bizonyos feltételnek eleget tevő tanácstagokat vesz figyelembe. A függvény visszatérési értéke `int`, és a feltételt, amelynek megfelelő tanácstagok számát visszaadja, egy delegate segítségével kapja meg paraméterként (tehát a `CountIf`-nek kell legyen paramétere).

> **Delegate típusa**
>
> A delegate típusa kötelezően a beépített generikus `Action` / `Func` delegate típusok közül a megfelelő kell legyen (vagyis saját delegate típus, ill. a beépített `Predicate` típus nem használható).
>
> Emiatt a listán NEM használhatod a beépített `FindAll` műveletét, mivel az általunk használt delegate típus nem lenne kompatibilis a `FindAll` által várt paraméterrel. A tagokon egy `foreach` ciklusban végigiterálva dolgozz!

- A property és a függvény működését demonstráld egy erre dedikált közös függvényben, amit lásd el a `[Description("Feladat7")]` 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.

> **Fontos**
>
> A `[Description("Feladat7")]` attribútum csak egyetlen függvény fölött szerepelhet.

Megoldás

- A `Count` nevű property esetében csak a `get` ágnak van értelme, ezért a `set` ágat 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.
  - A `CountIf` függvény a feltételt paraméterként egy `bool Függvénynév(Jedi jedi)`

szignatúrájú szűrőfüggvényt várjon.

Beadás

Ellenőrzőlista ismétlésképpen:

- 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.
- Amíg nem vagy rutinos a Visual Studio Git szolgáltatásainak használatában, a push-t követően (legkésőbb akkor, amikor a házi feladatot beadottnak tekintjük) célszerű ellenőrizni a GitHub webes felületén a repository-ban a fájlokra való rápillantással, hogy valóban minden változtatást feltöltöttél-e.
- A GitHub felületén ellenőrizd a push-t követően, hogy a GitHub Action alapú előellenőrző hiba nélkül lefutott-e.
- 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.
- Természetesen saját munkát kell beadni (hiszen értékelésre kerül).

2024-04-18 Szerzők