.NET FEJLESZTÉS C# NYELVEN

7. forduló

RENDELKEZÉSRE ÁLLÓ IDŐ:

45:00

Ismertető a feladathoz

Sikeresen szinkronba hoztátok a két rendszert.

A hajó tökéletesen funkcionál, viszont az utazás végét nem várhatjátok ki ébren, vissza kell jutnotok a hibernáló kamrába.

Viki felvetette, hogy át kellene néznetek a kódbázist, és készítenetek kellene egy figyelmeztető rendszert, amely automatikusan felébreszt titeket, ha újabb üzemzavar lépne fel a hajó rendszereiben. A fejlesztés folyamán az alábbi kérdések merülnek fel:

Felhasznált idő: 00:16/45:00 **Elért pontszám:** 0/12

1. feladat 0/5 pont

MulticastDelegate-ek esetén hogyan hívjuk meg az összes feliratkozót, ha a hibákat is le szeretnénk bennük kezelni?

Válasz

- A delegate hívása lekezeli az összes feliratkozó hibáját, tehát nincs szükség csak a hívásra magára.
- Egyszerűen a delegate hívását try/cath-be ágyazod.
- Kézzel váltod ki az eventeket használva a GetInvocationList-et külön lekezelve a hibákat.
- Ebben az esetben kerülendő az exception dobása és más módon kell a hibát lekezelni.

Magyarázat

Minden hibát (error-
t) egyesével tudsz kezelni és meg is tudsz bizonyosodni arról, hogy az összes feliratkozó meghívásra került.

A hívás try/catchbe ágyazásakor a programvégrehajtás megáll az első kivételnél, így a többi hiba nem kerül kezelésre.

Alapesetben a feliratkozók hívása nem folytatódik az első kezeletlen kivétel után.

A hibák kezelése exception-
nel előnyösebb. Eventek értékkel való visszatéréséhez így is szükség lesz az események egyenkénti hívásához.

2. feladat 0/7 pont

Az alábbi három metódus közül melyiknek tér vissza a leghosszabb stringgel?

```
public class Insane
{
    public record MyClass(string Z, string T, int Y);

    public string Case4()
    {
        var I = new MyClass("Forever", "foo", 11); int III = 3;
        var II = Enumerable.Repeat(I, III)
          .Select(i => i with { T = i.T + i.Y })
          .ToList().ToDictionary(_ => --III);
        var IIII = II.Select(p3 => p3.Value with { Y = p3.Key.ToString().Length });
        return IIII.Sum(p5 => p5.T.Length).ToString();
    }

    public string Case3()
    {
        var I = new MyClass("Asd", "foo", 11); int III = 1;
        var II = Enumerable.Repeat(I, III)
          .Select(i => i with { Y = ++III % 2 }).ToList()
          .ToDictionary(_ => III++);
        var charmander = II.GroupBy(a => a.Value);
        charmander.OrderByDescending(c => c.Key);
        return charmander.First().ToString();
    }

    public string Case2()
    {
        IEnumerable<(string N, int U)> list = new List<(string, int)>() {
          new("foo", 1), new("bur", 2), new("baz3", 3), new("foo12", 1),
          new("ba", 2), new("fu", 3), new("fubar", 4), new("bazbur", 5)
        };
        var test = list.Where(x => x.N.Length == list.Max(l => l.N.Length))
          .SelectMany(t => t.N)
          .Where(x => x.ToString().Length < list.Max(l => l.N.Length) - 1);
        return test.Last().ToString();
    }

    public string Case1()
    {
        IEnumerable<(string N, string U)> list = new List<(string, string)>() {
          new("foo", "1"), new("bur", "2"), new("baz3pikachu", "3"),
          new("foo12", "1"), new("ba", "2"), new("fu", "3"),
          new("fubar", "4"), new("bazbur", "5")
        };
        return list
          .GroupBy(x => x.N.Length == list.Max(l => l.N.Length) ? "x" : "y")
          .ToList().Last().Key;
    }
}
```

Válasz

- Case4();
- Case3();
- Case2();
- Case1();

Magyarázat

Case4(); Kimenete: 15

Case3(); Kimenete: "System.Linq.Grouping`2[TryMe.Insane+MyClass,System.Collections.Generic.KeyValuePair`2[System.Int32,TryMe

Case2(); Kimenete: "r"

Case1(); Kimenete: "x"