.NET FEJLESZTÉS C# NYELVEN

2. forduló

RENDELKEZÉSRE ÁLLÓ IDŐ:

15:00

Ismertető a feladathoz

Fontos információk

Ha kifutsz az adott feladatlap kitöltésére rendelkezésre álló időből, a felület **automatikusan megpróbálja beküldeni** az addig megadott válaszokat

A kérdésekre **mindig van helyes válasz**! Ha csak egy helyes válasz van az adott kérdésre, radio button-os választási lehetőségeket fogsz látni.

Olyan kérdés viszont nincs, amelyre az összes válasz helyes!

Egyéb információkat a versenyszabályzatban találsz!

Második forduló

Sikerült jogosultságot szerezni a tableten és pár próbálkozás után az ajtó vezérlőjéhez is csatlakoztál.

Itt az ideje elhagyni a gyengélkedőt.

Felhasznált idő: 00:13/15:00 **Elért pontszám:** 0/7

1. feladat 0/2 pont

A kiolvasztását követően pár órában nehézkes felidézni az emlékeidet.

Pár elgondolkodtató kérdés megválaszolásával lerövidítheted ezt az állapotot.

Egy osztályod két interfészt implementál IA és IB. Mindkettő interfészben van egy metódus ugyanazzal a névvel. Az IA interfészben található implementáció kell, hogy legyen az alapértelmezett, IB implementációját csak ritka esetekben szeretnéd használni. Hogy implementálod ezeket az interfészeket?

Válasz

- ○ IA-t implicitként és IB-t is implicitként.

- IA-t explicitként és IB-t is explicitként.
- IA-t implicitként és IB-t explicitként.
- IA-t explicitként és IB-t implicitként.

Magyarázat

Az IA-t implicitként való implementálásával az IA lesz az alapértelmezett működés, ritka esetben az IB explicitként implementálható.

2. feladat 0/3 pont

Az ajtó nyitásához az alábbi két kódrészlet megfejtésére lesz szükséged.

Mi lesz GetMax() hívás eredménye? (10 < _options.Value.End < 100)

```
public class YieldReturn
{
    private readonly IOptions<Options> _options;

    public YieldReturn(IOptions<Options> options)
    {
        _options = options;
    }

    public int GetMax()
    {
        IEnumerable<int> enumeration;
        try
        {
            enumeration = FillEnumeration(1);
        }
        catch (InvalidOperationException)
        {
            enumeration = FillEnumeration(_options.Value.End);
        }
        return enumeration.Max();
    }

    public IEnumerable<int> FillEnumeration(int end)
    {
        if (end == 1)
          throw new InvalidOperationException("Max should be bigger then 1");

        for (int i = 0; i <= end; i++)
        {
            yield return i;
        }
    }
}
```

Válasz

- 1
- _options.Value.End
- _options.Value.End + 1
- egyéb

Magyarázat

A FillEnumeration(1) a try catch blockon kívül található enumeration.Max() hívással kerül kiértékelésre.

3. feladat 0/2 pont

Mit ír ki az alábbi kód?

```
public class App
{
    delegate int NumDelegate();
    public void Calc()
    {
        NumDelegate number1 = () => int.MaxValue;
        NumDelegate number2 = () => 1;
        NumDelegate sum = () => number1() + number2();
        Console.WriteLine(sum);
    }
}
```

Válasz

- -1
- 1
- -2147483648
- egyéb
- exception történik

Magyarázat

A Console.WriteLine(sum) nem lefuttatja, hanem meghívja a sum ToString metódusát.