.NET FEJLESZTÉS C# NYELVEN

1. forduló

DXC TECHNOLOGY

A kategória támogatója: DXC Technology

Ismertető a feladathoz

Kérjük, hogy a feladatlap indítása előtt mindenképp olvasd el az alábbi útmutatót:

MINDEN kérdésre **van helyes válasz**.

Olyan kérdés **NINCS**, amire az összes válasz helyes, ha mégis az összes választ bejelölöd, arra a feladatra automatikusan 0 pont jár.

A **radio button-os kérdésekre egy helyes válasz van**.

Ha lejár a feladatlap ideje, a rendszer AUTOMATIKUSAN beküldi azt az addig megjelölt válaszokkal.

Azokat a feladatlapokat, amelyekhez **csatolmány** tartozik, javasoljuk **NEM mobilon** elindítani, erre az érintett feladatlapok előtt külön felhívjuk a figyelmet.

Az **adatbekérős feladatokra NEM jár részpontszám**, csak a feleletválasztósakra.

Helyezéseket a 4. forduló után mutatunk, százalékos formában: adott kategóriában a TOP 20-40-60%-hoz tartozol.

Badge-ket szintén a 4.forduló után kapsz majd először.

Ugrás a tartalomhoz

- Ha egyszerre több böngészőből, több ablakban vagy több eszközről megnyitod ugyanazt a feladatlapot, **nem tudjuk vállalni** az adatmentéssel kapcsolatban

esetlegesen felmerülő anomáliákért a felelősséget!

A hét forduló során az egyes kategóriákban (de nem feltétlenül mindegyikben) **könnyű-közepes-nehéz kérdésekkel** egyaránt találkozhatsz majd.

Jó versenyzést kívánunk!

Felhasznált idő: 00:00/10:00 **Elért pontszám:** 0/15

1. feladat 0/5 pont

Mi a következő kód kimenete?

```
var numbers = new List<int>();
int? n = null;

numbers.Add(n ??= 17);
numbers.Add(n ??= 20);

Console.WriteLine(string.Join(" ", numbers));
```

Válasz

A helyes válasz:

17 17

Ugrás a tartalomhoz

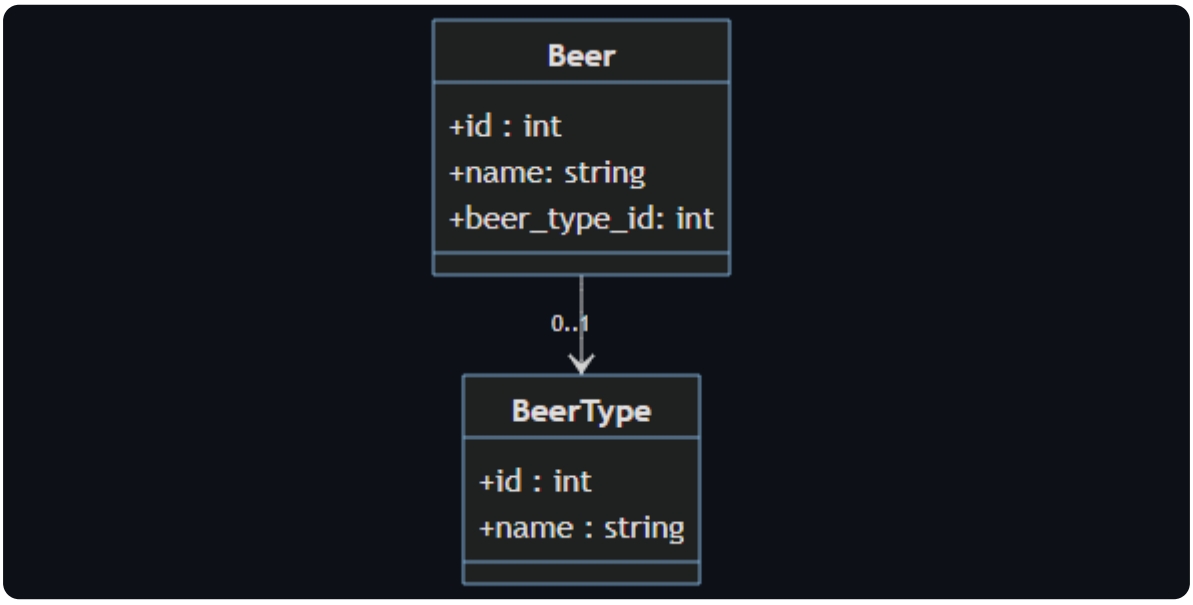
Magyarázat

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/null-coalescing-operator

2. feladat 0/10 pont

Vizsgáld meg az alábbi kódrészleteket, és jelöld meg a **helyes** álltásokat!

A feladat során feltételezheted, hogy a program EF Core 6-ot használ és az alábbi táblarelációt valósítja meg.



```
public IEnumerable<Beer> GetBeers(BeerDataContext
context)
{
    var lagerTypes = new string[]
    {
        BeerTypeEnum.Lager,
        BeerTypeEnum.Pilsner
    };
    return context.Beers.Where(b =>
        b.BeerType != null
        && lagerTypes.Contains(b.BeerType.Name));
}
```

```
public IQueryable<Beer> GetLagers(BeerDataContext
context)
{
    return context.Beers
        .Include(b => b.BeerType)
        .Where(b => IsLager(b.BeerType));
}

private bool IsLager(BeerType bt)
{
    if (bt == null) return false;
    var lagerTypes = new string[]
    {
        BeerTypeEnum.Lager,
        BeerTypeEnum.Pilsner
    };
    return lagerTypes.Contains(bt.Name);
}
```

Ugrás a tartalomhoz

- A két függvény működésében nincs különbség.

A GetLagers függvény *System.InvalidOperationException*-t dob, mert a LINQ nem tudja lefordítani.

A GetBeers függvény gyorsabb, mert kizárólag *CLR* függvényeket használ.

Magyarázat

EF LINQ csak CLR metódusokat tartalmazhat, GetBeers-ben használt IsLager metódus nem CLR, ezért az dobja futási időben az InvalidOperationException-t

https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/language-reference/clr-method-to-canonical-function-mapping