Haladó Programozás I: minta ZH

Alapkód

Készítsen egy konzolos applikációt .NET 8 keretrendszer használatával. A program egy horgászbolt webshop eladásait kezeli, amely termékeket, ügyfeleket és rendeléseket kezel.

Entitások:

1. **Customer osztály:**
    - Id (int)
    - Name (string)
    - Email (string)
2. **Order osztály:**
    - Id (int)
    - Date (DateTime)
3. **Product osztály:**
    - Id (int)
    - Name (string)
    - Category (string)
    - Price (double)

Kapcsolatok:

- Minden Order-hez tartozik 1db Customer, 1 Customerhez több Order.
- A Productok és az Orderek között több-több kapcsolat van, viszont meg kell tudni adni hogy adott Orderhez adott Productból hány db tartozik.

A feladatban mindhárom entitáshoz külön manager osztály tartozik, az egyes megoldásokat az entitásnak megfelelő manager osztályban valósítsd meg.

Aláírás szint:

1) A következő feladatok mindegyikét kötelezően meg kell oldani, ezek szükségesek a további feladatok megoldásához.
    a) Implementáld a Product és Customer entitásokhoz a Create (létrehozás) és a GetAll (listázás) függvényeket. Figyelj, hogy a megadott Main függvény le tudjon futni! 10 pont
    b) Készítsd el az első migrációt és azt alkalmazd az adatbázisodon. 5 pont
    c) A letöltött InitDbData.sql fájl alapján töltsd fel az adatbázisod példaadatokkal. 3 pont
2) Készíts egy függvényt mely visszaadja, hogy melyik nap hány rendelés történt. A függvényt az OrderManager osztályban implementáld, a Mainben hívd meg, és az eredményt a konzolra írasd ki. 4 pont
3) Készíts egy függvényt, mellyel rendeléseket lehet felvinni és hozz létre 3 rendelést. Ehhez használd a példa adatokat, minden rendelésnek legyen vásárlója, dátuma és legalább 1 megrendelt terméke. A rendelést létrehozó függvényt az OrderManager osztályban implementáld és a Main függvényben hívd meg a Product és Customer mintájára. 3 pont

4) Egészítsd ki úgy a megrendelés felvétele függvényt, hogy, ha a megrendelő szerepel a tiltólistán, mely a blackList.json file-ban található és megrendelést akarnánk felvinni hozzá akkor dobjon egy saját készítésű Exception-t a program, mely tartalmazza a megrendelő nevét. A Main függvényben demonstráld a működést. 4 pont

5) Készíts egy lekérést mely visszaadja minden termékre, hogy azokat eddig milyen megrendelők rendelték, az egyes termékeknél minden hozzá tartozó megrendelő csak egyszer szerepeljen. A függvényt a ProductManager osztályban implementáld. A működést demonstráld a Main függvényben, az eredményt írasd ki a konzolra. 3 pont

6) Egészítsd ki a megrendelő entitást egy TotalOrders (int) property-vel, mely tárolja a megrendelőhöz tartozó eddig összes megrendelés számát. Készíts migrációt, alkalmazd az adatbázison. 2 pont

7) Készíts egy függvényt, mely kiszámolja az összes eddigi megrendelőre az adatbázisban a megrendelések össz számát és elmenti azt. A függvényt a CustomerManager osztályban implementáld. Módosítsd a megrendelés felvétele függvényt úgy, hogy minden megrendelésnél frissítse a megrendelő megrendelésszámát. Írasd ki a konzolra a megrendelők listáját úgy, hogy már a megrendeléseik száma is benne legyen. 5 pont

8) Készíts egy új entitást és valósítsd meg hozzá az alábbi feladatokat.

   a) **Address:**

      - Id (int)
      - ZipCode (int)
      - City (string)
      - Street (string)

   b) Minden Customer-hez tartozik 1db Address.

   c) Konfiguráld az új entitást a következő módon 4 pont

      - A ZipCode default értéke legyen 8200
      - A City default értéke legyen Veszprém
      - Az Street default értéke legyen Egyetem u. 10.
      - Minden property legyen kötelező az Addresses táblában

   d) Készíts migrációt, alkalmazd az adatbázison és a letöltött Addresses.sql fájl alapján töltsd fel az adatbázisod példa adatokkal. 2 pont

9) Készíts egy lekérdezést, amely kiszámolja, hogy különböző városokban (City) hány rendelés történt, városok neve alapján rendezve csökkenő sorrendben. Az eredmény adatok: *Város neve, Rendelések száma*. Az eredményt a program egy új adatbázis táblába írja ki. A függvényt az OrderManager osztályban implementáld. 5 pont

LINQ függvények segédlet

A feladatok megoldásához ad kis segítséget, nem kizárólag ezeket a függvényeket lehet használni a megoldás során.

1. **All**: Ellenőrzi, hogy a gyűjtemény összes eleme megfelel-e a megadott feltételnek
2. **Any**: Ellenőrzi, hogy van-e legalább egy elem a gyűjteményben, amely megfelel a megadott feltételnek
3. **Average**: Visszaadja a gyűjtemény elemeinek átlagát
4. **Count**: Visszaadja a gyűjtemény elemeinek számát
5. **Distinct**: Visszaadja a gyűjtemény egyedi elemeit, eltávolítva a duplikátumokat
6. **First**: Visszaadja a gyűjtemény első elemét
7. **FirstOrDefault**: Visszaadja a gyűjtemény első elemét, vagy az alapértelmezett értéket, ha a gyűjtemény üres
8. **GroupBy**: Csoportosítja a gyűjtemény elemeit egy kulcs alapján
9. **Join**: Két gyűjteményt összekapcsol a közös kulcs alapján, és új elemeket hoz létre, amelyek tartalmazzák a két gyűjtemény elemeit
10. **Last**: Visszaadja a gyűjtemény utolsó elemét
11. **LastOrDefault**: Visszaadja a gyűjtemény utolsó elemét, vagy az alapértelmezett értéket, ha a gyűjtemény üres
12. **Max**: Visszaadja a gyűjtemény legnagyobb elemét
13. **MaxBy**: Visszaadja a gyűjtemény azon elemét, amely a megadott kulcs alapján a legnagyobb értékkel bír
14. **Min**: Visszaadja a gyűjtemény legkisebb elemét
15. **MinBy**: Visszaadja a gyűjtemény azon elemét, amely a megadott kulcs alapján a legkisebb értékkel bír
16. **OrderBy**: Rendez egy gyűjteményt egy megadott kulcs alapján növekvő sorrendben
17. **OrderByDescending**: Rendez egy gyűjteményt egy megadott kulcs alapján csökkenő sorrendben
18. **Select**: Átalakítja a gyűjtemény elemeit egy új formátumra
19. **SelectMany**: Kiterjeszti a gyűjtemény elemeit, és laposítja az eredményt, így egy tömb vagy lista elemeit egyesíti
20. **Single**: Visszaadja a gyűjtemény egyetlen elemét, ha csak egy van, különben kivételt dob
21. **SingleOrDefault**: Visszaadja a gyűjtemény egyetlen elemét, vagy az alapértelmezett értéket, ha nincs vagy több mint egy elem van
22. **Sum**: Visszaadja a gyűjtemény elemeinek összegét
23. **ToDictionary**: Átalakítja a gyűjteményt egy szótárrá, ahol a kulcsokat és értékeket a megadott kulcsgeneráló és értékgeneráló függvények határozzák meg
24. **ToList**: Átalakítja a gyűjteményt egy List<T> típusú gyűjteménnyé
25. **Where**: Szűri a gyűjtemény elemeit egy megadott feltétel alapján
26. **Zip**: Két gyűjteményt összekapcsol, és egy új gyűjteményt hoz létre az elemek párba állításával

Migrációk használatához segédlet

Add-Migration: Új migráció hozzáadása.

- Létrehoz egy új migrációs osztályt, amely tartalmazza az aktuális adatmodell és az adatbázis közötti különbségeket.
- **Szintaxis (Package Manager Console):**

  Add-Migration MigrationName

- **Szintaxis (Console):**

  dotnet ef migrations add MigrationName

Update-Database: Az adatbázis frissítése a legutóbbi migrációk alapján.

- Alkalmazza az összes függőben lévő migrációt az adatbázisra.
- **Szintaxis (Package Manager Console):**

  Update-Database

- **Szintaxis (Console):**

  dotnet ef database update

Remove-Migration: Az utolsó migráció visszavonása (csak ha még nincs alkalmazva az adatbázisban).

- Törli az utoljára létrehozott migrációs osztályt.
- **Szintaxis (Package Manager Console):**

  Remove-Migration

- **Szintaxis (Console):**

  dotnet ef migrations remove

Ef Core függvények segédlet

Include: Kapcsolt entitások betöltése egy együtt-kapcsolatban (lazy loading helyett explicit).

ThenInclude: Többszintű navigációs tulajdonságok betöltése (pl. egy kapcsolat további mélységei).

SaveChanges: Az aktuális változtatások (pl. hozzáadás, frissítés, törlés) mentése az adatbázisba.

Add: Egy új entitás hozzáadása az adott DbContext-hez (INSERT művelet az adatbázisban).

Update: Egy meglévő entitás állapotának módosítása (UPDATE művelet az adatbázisban).

Remove: Egy entitás eltávolítása az adatbázisból (DELETE művelet).