

# JEGYZŐKÖNYV

## PROGRAMOZÁS AZ ENTITY FRAMEWORK HASZNÁLATÁVAL

GYAKORLAT ADATOK	
NÉV	CSATLÓS TAMÁS PÉTER
NEPTUN KÓD	TALXCE
HALLGATÓ TANSZÉKE/SZAKIRÁNYA	IIT, RENDSZERFEJLESZTŐ
GYAKORLATVEZETŐ	
GYAKORLAT HELYE	IL206
GYAKORLAT IDŐPONTJA	2016.04.18.
GYAKORLAT SORSZÁMA	2.

FELADATLISTA	
1. FELADAT	<input type="checkbox"/>
2. FELADAT	<input type="checkbox"/>
3. FELADAT	<input type="checkbox"/>
4. FELADAT	<input type="checkbox"/>
5. FELADAT	<input type="checkbox"/>
6. FELADAT	<input type="checkbox"/>
7. FELADAT	<input type="checkbox"/>
9. FELADAT	<input type="checkbox"/>
10. FELADAT	<input type="checkbox"/>
11. FELADAT	<input type="checkbox"/>
12. FELADAT	<input type="checkbox"/>
14. FELADAT	<input type="checkbox"/>

## FELADATOK

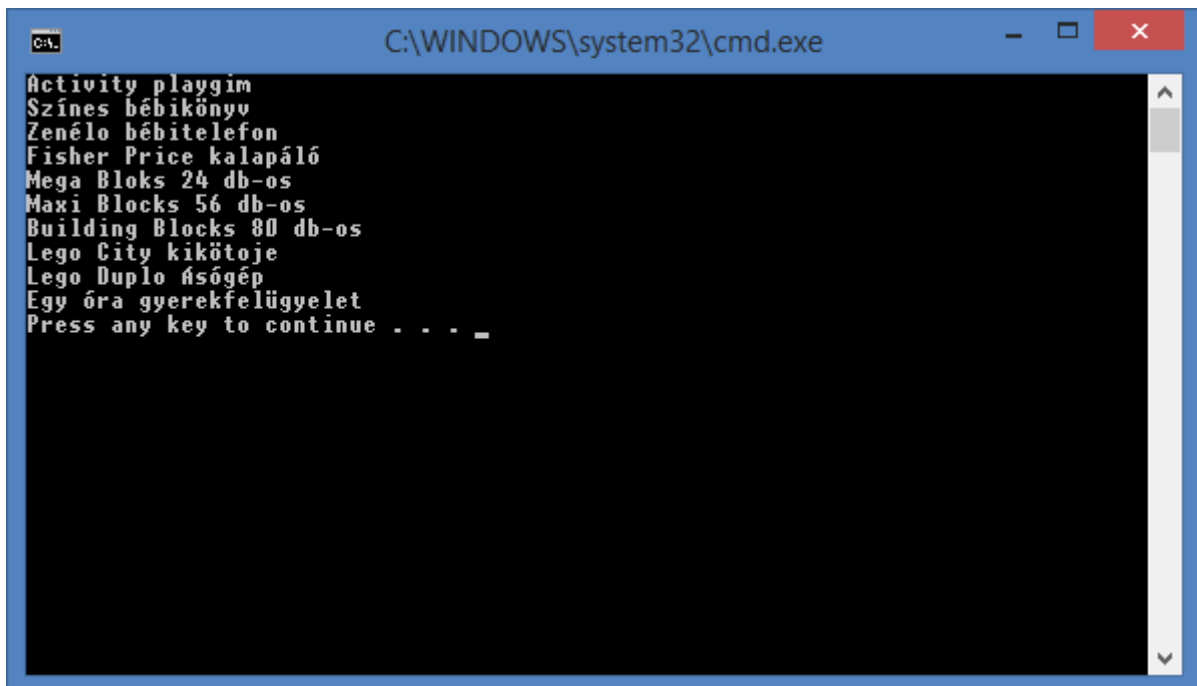
### 2.FELADAT

#### Feladat neve és leírása:

Készíts egy függvényt TermekListazas() néven, mely kiírja a termékek neveit.

#### Megoldás:

```
static void TermekListazas()
{
    using (var context = new LABOREntities())
    {
        foreach (var termek in context.Termek)
        {
            Console.WriteLine("{0}", termek.Nev);
        }
    }
    //Lezarja a kapcsolatot
}
```



#### Indoklás:

Az adatbázis kapcsolatot egy LABOREntities típusú objektum reprezentálja. Using blokkban használva biztosított a kapcsolat lezárása. A függvény kiírja az összes termék nevét. Képesek vagyunk egy egyszerű ciklussal lekérni az összes rekordot és azon belül a név attribútumot. Először a foreach első iterációjakor nyúlunk az adatbázishoz a Lazy loading miatt. Megfigyelhető az is, hogy a táblát úgy használom mint egy kollekció. Egy rekord ebből a kollekcióból egy elem, és elérem az attribútumait mint tagváltozók.

### 3.FELADAT

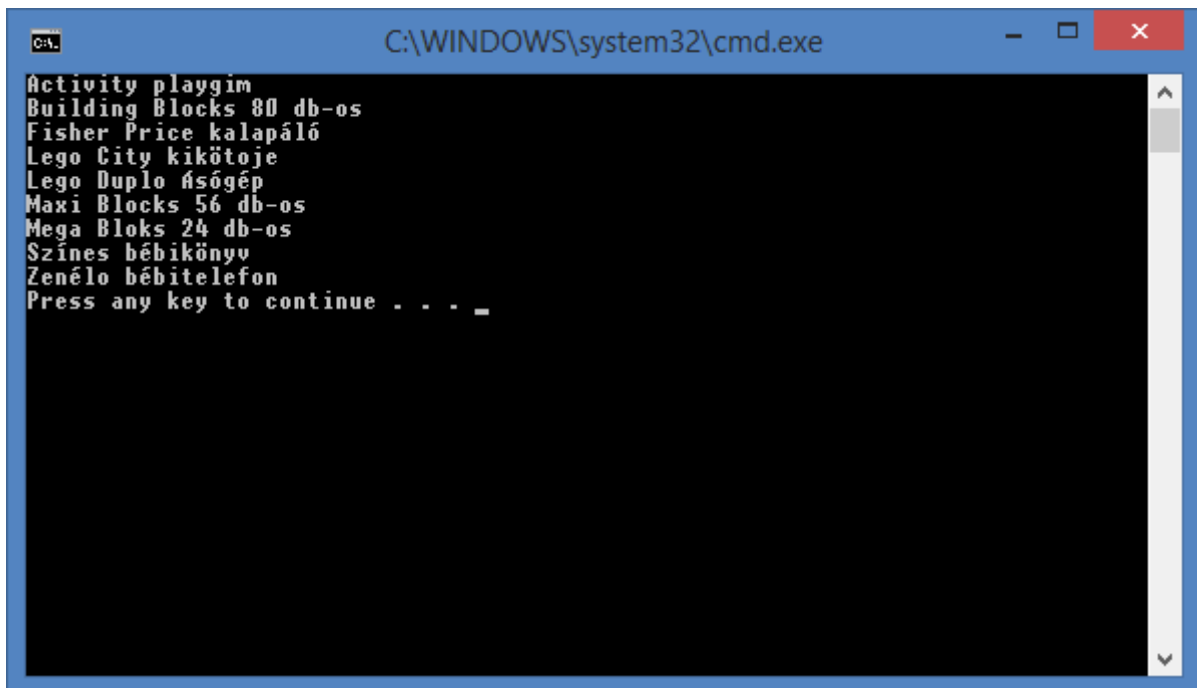
#### Feladat neve és leírása:

Készíts egy előzőhöz hasonló függvényt `HatekonyTermekListazas()` néven, mely csak a készleten lévő termékeket írja ki, a termékeket betűrendbe rendezi, illetve hogy az adatbázisból csak a megjelenítendő adatokat kéri le!

#### Megoldás:

```
static void HatekonyTermekListazas()
{
    using (var context = new LABOREntities())
    {
        IOrderedQueryable<Termek> query = from t in context.Termek
                                           where t.Raktarkeszlet > 0
                                           orderby t.Nev
                                           select t;

        foreach (var termék in query)
        {
            Console.WriteLine("{0}", termék.Nev);
        }
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
Activity playgim
Building Blocks 80 db-os
Fisher Price kalapáló
Lego City kikötője
Lego Duplo ásógép
Maxi Blocks 56 db-os
Mega Bloks 24 db-os
Színes bábikönyv
Zenélo bábitelefon
Press any key to continue . . . _
```

#### Indoklás:

Ebben a feladatban használtam a Linq által nyújtott másik módszert az adatok eléréséhez és szűréséhez. A query lekérdezés adatbázis oldalon egy SQL utasításként fog lefutni. A típusa a kódban látható és leszármazik az `IEnumerable` interfészből is.

#### 4.FELADAT

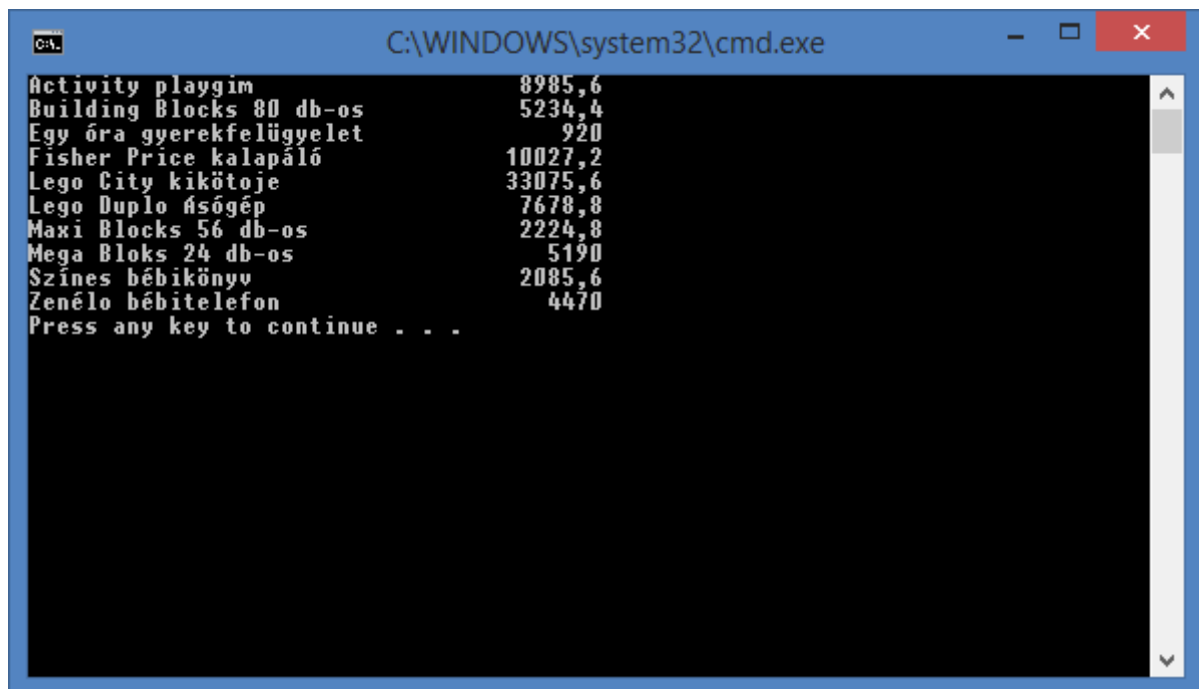
##### Feladat neve és leírása:

Készíts egy függvényt TermekListazasBrutto() néven mely a termékek neveit és bruttó árait írja ki.

##### Megoldás:

```
static void TermekListazasBrutto()
{
    using (var context = new LABOREntities())
    {
        var query = from t in context.Termek
                     orderby t.Nev
                     select new
                     {
                         Nev = t.Nev,
                         Brutto = t.NettoAr * (100 + t.AFA.Kulcs) / 100
                     };

        foreach (var termek in query)
        {
            Console.WriteLine("{0, -25} {1, 15}", termek.Nev, termek.Brutto);
        }
    }
}
```



Termék neve	Bruttó ár
Activity playgim	8985,6
Building Blocks 80 db-os	5234,4
Egy óra gyerekfelügyelet	920
Fisher Price kalapáló	10027,2
Lego City kikötője	33075,6
Lego Duplo ársógép	7678,8
Maxi Blocks 56 db-os	2224,8
Mega Bloks 24 db-os	5190
Színes bébikönyv	2085,6
Zenélo bébitelefon	4470

##### Indoklás:

Feladat megoldásához egy anonim osztályt használtam. Ebben két tagváltozót hozok létre és ezeket felhasználom a kiíratásnál. Ki tudom számolni a bruttó árát egy terméknek, úgy hogy az attribútumait használom a fent látható módon.

## 5.FELADAT

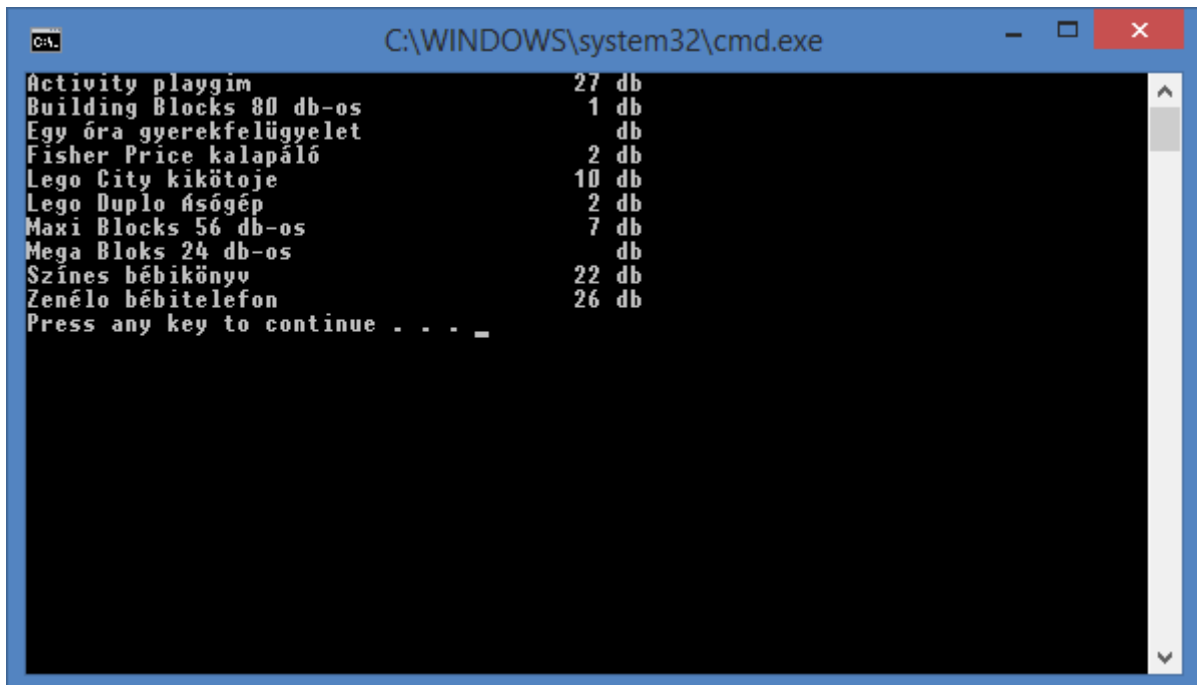
### Feladat neve és leírása:

Készíts egy függvényt Megrendelesek() néven, mely felsorolja a termékeket és minden termék mellett feltünteti a megrendelések összegét! Ügyelj rá, hogy a listázás egyetlen SQL utasítás végrehajtásával történjen

### Megoldás:

```
static void Megrendelesek()
{
    using (var context = new LABOREntities())
    {
        var query = from t in context.Termek
                     orderby t.Nev
                     select new
                     {
                         Nev = t.Nev,
                         Ossz = t.MegrendelesTetel.Sum(m => m.Mennyiseg)
                     };

        foreach (var termek in query)
        {
            Console.WriteLine("{0, -25} {1, 15} db", termek.Nev, termek.Ossz);
        }
    }
}
```



Termék	Ossz
Activity playgim	27 db
Building Blocks 80 db-os	1 db
Egy óra gyerekefelügyelet	db
Fisher Price kalapáló	2 db
Lego City kikötője	10 db
Lego Duplo Ásógép	2 db
Maxi Blocks 56 db-os	7 db
Mega Bloks 24 db-os	db
Színes bábikönyv	22 db
Zenéző bábitelefon	26 db

### Indoklás:

Mivel az IEnumerable interfészből származik a query típusa, ezért alkalmazhatóak a lambda kifejezések is.

## 6.FELADAT

### Feladat neve és leírása:

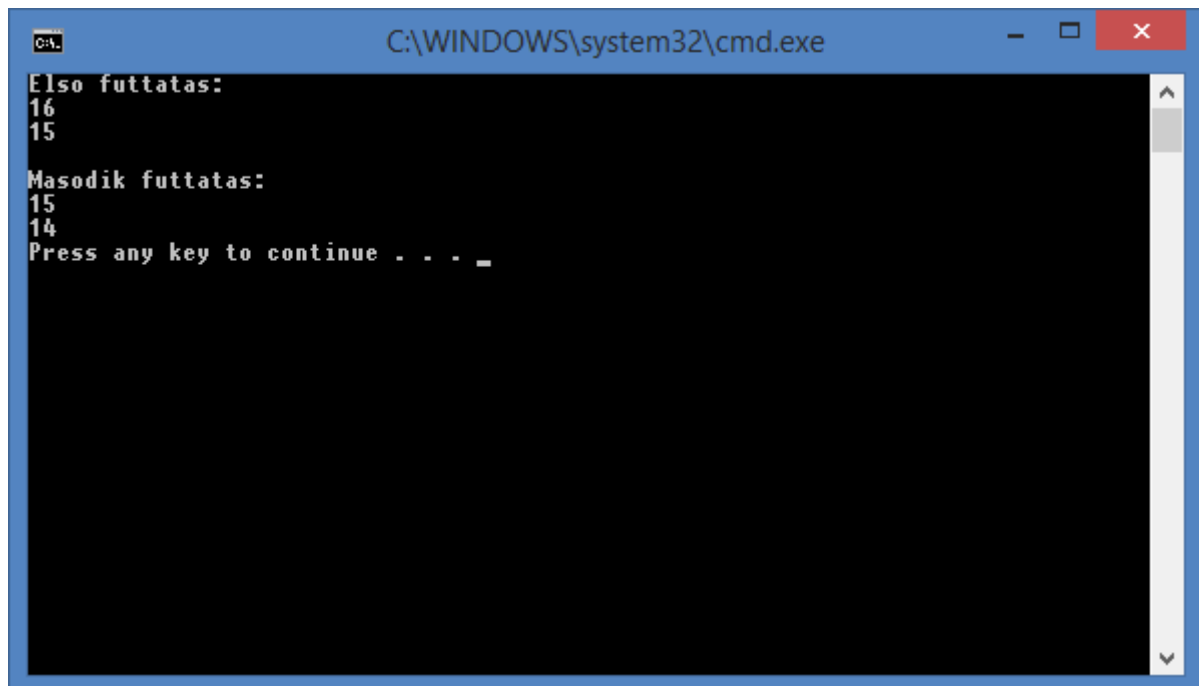
Készíts függvényt Eladas() néven mely a 3-as azonosítójú termék készletét 1-gyel csökkenti, majd kiírja az új készletet.

### Megoldás:

```
static void Eladas()
{
    using (var context = new LABOREntities())
    {
        //SQL: update utasítás lenne
        Termek termék = context.Termek.Single(t => t.ID == 3);

        Console.WriteLine(termek.Raktarkeszlet);
        termék.Raktarkeszlet--;
        Console.WriteLine(termek.Raktarkeszlet);

        context.SaveChanges();// explicit nekünk kell megadni!
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
Első futtatás:
16
15

Második futtatás:
15
14
Press any key to continue . . . _
```

### Indoklás:

Feladathoz kétszer egymás után hívtam meg a függvényt és látható, hogy a raktárkészlet csökken minden hívás után. SQL utasításként ez UPDATE parancsnak felel meg. A végén nekünk kell gondoskodni a változások mentéséről. Továbbá a Single() metódus hibát dob, ha nincs mivel visszatérnie.

## 7.FELADAT

### Feladat neve és leírása:

Készíts egy függvényt Beszallitas() néven, mely felvesz egy új kategóriát és egy abba tartozó új terméket az adatbázisba, végül pedig kiírja az új sorok azonosítóit. A művelet során ne használjuk ki, hogy ismerhetjük az új kategória azonosítóját

### Megoldás:

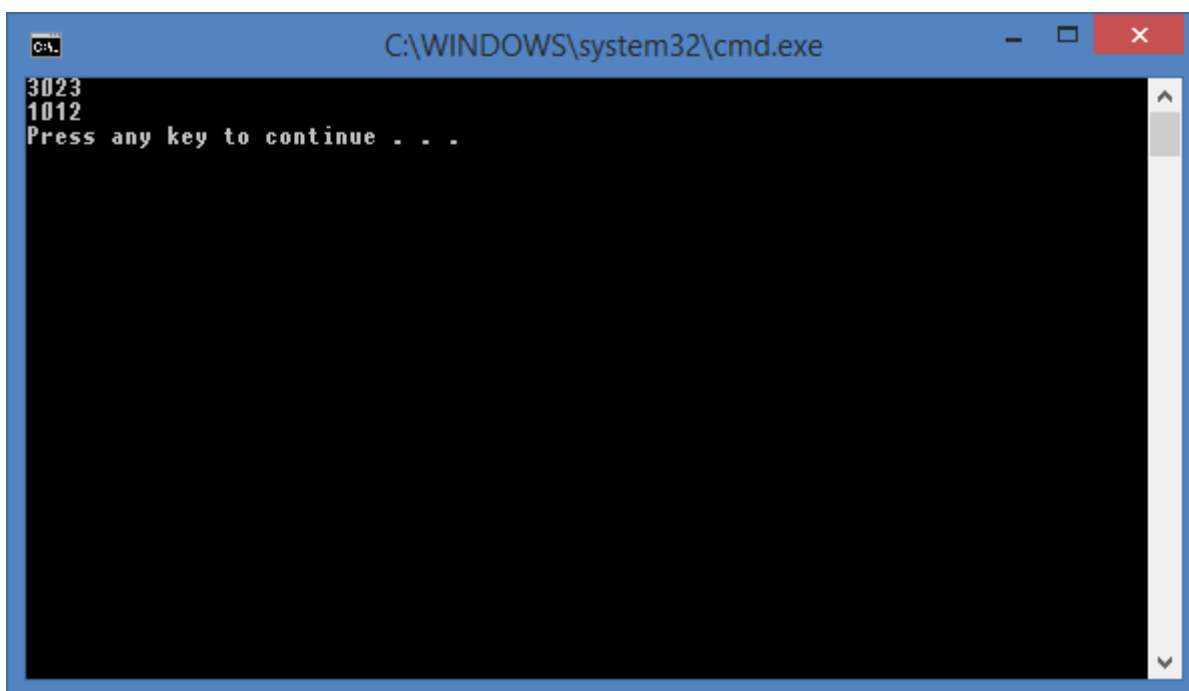
```
static void Beszallitas()
{
    using (var context = new LABOREntities())
    {
        Katategoria kategoria = new Katategoria();
        kategoria.Nev = "Csonthejasok";

        Termek termék = new Termek();
        termék.Nev = "Nagy kakas";
        termék.NettoAr = 420;

        termék.Kategoria = kategoria;

        context.Termek.Add(termék);
        context.SaveChanges();

        Console.WriteLine(kategoria.ID);
        Console.WriteLine(termék.ID);
    }
}
```



```
C:\WINDOWS\system32\cmd.exe
3023
1012
Press any key to continue . . .
```

### Indoklás:

Beszúrás és az adatok egyéb módú módosítása nem nehéz feladat az EF segítségével. A kategóriát el lehetne külön is menteni, de a navigációs property-k miatt elég értékül adni a terméknek és a terméket visszaírni az adatbázisba.

## 8.FELADAT

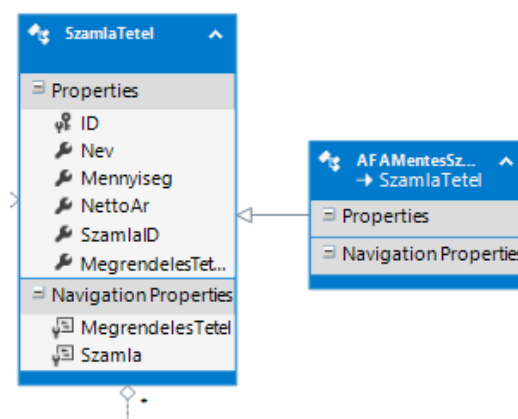
### Feladat neve és leírása:

Az Entity Framework segítségével válaszd szét az ÁFA mentes számlatételt a többitől osztály szinten is. Ennek érdekében vezess be egy AfaMentesSzamlaTetel entitástípust, mely a SzamlaTetel típussal származási relációban van! Megoldásod tesztelésére készíts egy függvényt AfaMentesTetelek() néven, mely kilistázza az összes számlatételt, megjelölve azok osztályát! (Tipp: az alapértelmezett adatbázisban nincs 1-es kódú tétel, a megfelelő teszteléshez új számlatételeket kell felvenned, vagy meglévőket módosítani. Ezt megteheted közvetlenül az SQL Server Management Konzol használatával is.)

### Megoldás:

```
static void AfaMentesTetelek()
{
    using (var context = new LABOREntities())
    {
        foreach (var tetel in context.SzamlaTetel)
        {
            Console.WriteLine("{0}\t{1}", tetel.Nev, tetel.GetType());
        }
    }
}
```

	ID	Nev	Mennyiség	NettoAr	AFAKulcs	SzamlalD	MegrendelesT...
	1	Fisher Price kal...	2	8356	20	1	1
	2	Maxi Blocks 56 ...	1	1854	20	1	2
	3	Színes bébikönyv	5	1738	20	1	3
	4	Activity playgym	2	7488	20	2	4
	5	Zenélo bébitele...	3	3725	20	2	5
	6	Teszt	2	200	0	1	6
➤*	NULL	NULL	NULL	NULL	NULL	NULL	NULL





Column	Operator	Value / Property
Tables		
Maps to SzamlaTetel		
When AFAKulcs	=	0
<Add a Condition>		
Column Mappings		
AFAKulcs : int	↔	🔧
<Add a Table or View>		

```

Select C:\Windows\system32\cmd.exe
Fisher Price kalapáld System.Data.Entity.DynamicProxies.SzamlaTetel_FDA98F82D844D95F243B4D03162C8B7D8F25620AE2E77F2E5C
70F41172DE94D8
Maxi Blocks 56 db-os System.Data.Entity.DynamicProxies.SzamlaTetel_FDA98F82D844D95F243B4D03162C8B7D8F25620AE2E77F2E5C
70F41172DE94D8
Színes bábikönyv System.Data.Entity.DynamicProxies.SzamlaTetel_FDA98F82D844D95F243B4D03162C8B7D8F25620AE2E77F2E5C
70F41172DE94D8
Activity playgym System.Data.Entity.DynamicProxies.SzamlaTetel_FDA98F82D844D95F243B4D03162C8B7D8F25620AE2E77F2E5C
70F41172DE94D8
Zenőlo bébitelefon System.Data.Entity.DynamicProxies.SzamlaTetel_FDA98F82D844D95F243B4D03162C8B7D8F25620AE2E77F2E5C
70F41172DE94D8
Teszt System.Data.Entity.DynamicProxies.AfaMentesSzamlaTetel_6790214A6F13D804993E1FA4A392980467620E9864922AC374CA411B1
1F170E7
Press any key to continue . . .

```

## Indoklás:

A feladat megoldásához az első képen látható Teszt nevű rekordot szűrtem be a számla tétel táblába (1.kép). Továbbá leszármaztattam a Számla Tétel entitásból egy Áfa mentes számla tétel entitást (2.kép) és beállítottam a leképezést is (3. kép). Végül futattam a metódust és látható, hogy a Teszt nevű Számla Tétel típusa más, mint a többi (4.kép).

## 9.FELADAT

### Feladat neve és leírása:

Készíts egy függvényt VevoListazas() néven, mely kiírja a vevők neveit

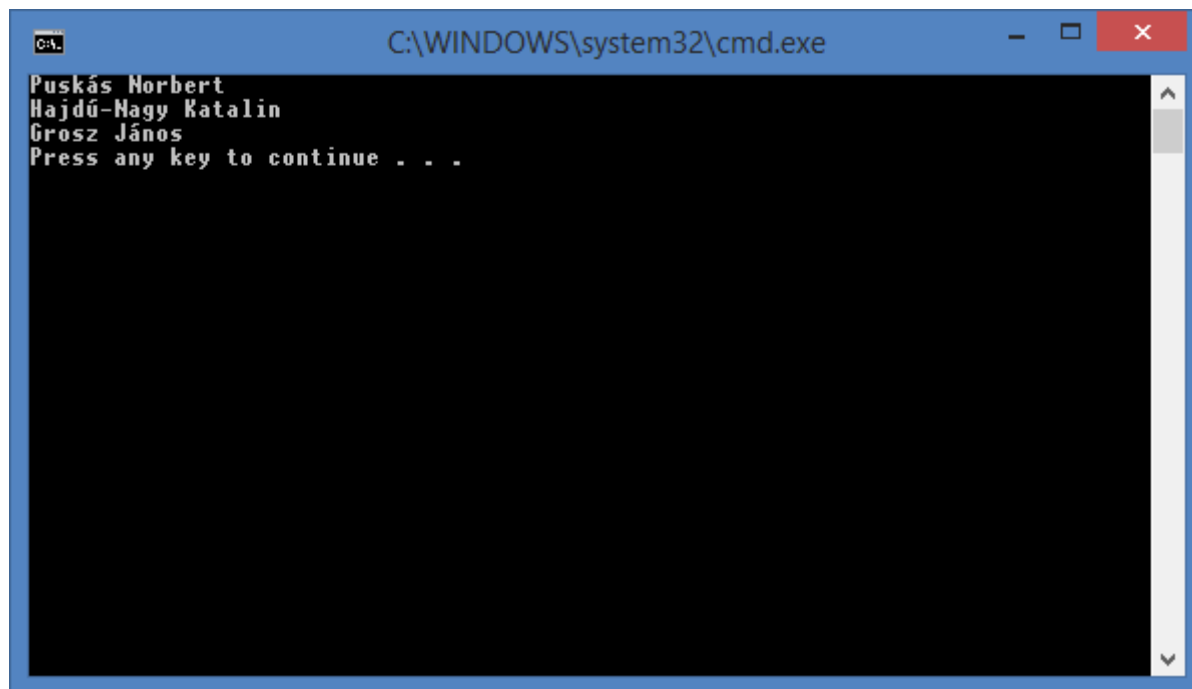
### Megoldás:

```

static void VevoListazas()
{
    using (var context = new LABOREntities())
    {
        foreach (var vevo in context.Vevo)
        {
            Console.WriteLine("{0}", vevo.Nev);
        }
    }
}

```

```
}  
  
}
```



#### Indoklás:

A 2.feladathoz nagyon hasonló ez a feladat, csak a vevőket kellett kilistázni.

### 10.FELADAT

#### Feladat neve és leírása:

Készíts egy előzőhöz hasonló függvényt `HatekonyVevoListazas()` néven, mely csak az „o” betűt tartalmazó nevű vevőket listázza ki, a vevőket betűrendbe rendezi, illetve az adatbázisból csak a megjelenítendő adatokat kéri le.

#### Megoldás:

```
static void HatekonyVevoListazas()  
{  
    using (var context = new LABOREntities())  
    {  
        var query = from v in context.Vevo  
                     where v.Nev.Contains("o")  
                     orderby v.Nev  
                     select v.Nev;  
  
        foreach (var vevo in query)  
        {  
            Console.WriteLine("{0}", vevo);  
        }  
    }  
}
```

```
}

C:\WINDOWS\system32\cmd.exe
Puskás Norbert
Hajdú-Nagy Katalin
Grosz János
-----
Grosz János
Puskás Norbert
Press any key to continue . . . _
```

#### Indoklás:

A lekérdezés where feltételében vizsgálom, hogy van-e a névben „o” betű. A lekérdezés az „o” betűt tartalmazó neveket adja vissza. Összehasonlításképp lefuttattam a vevők listázását is, hogy ellenőrizzem a működését a függvénynek.

### 11.FELADAT

#### Feladat neve és leírása:

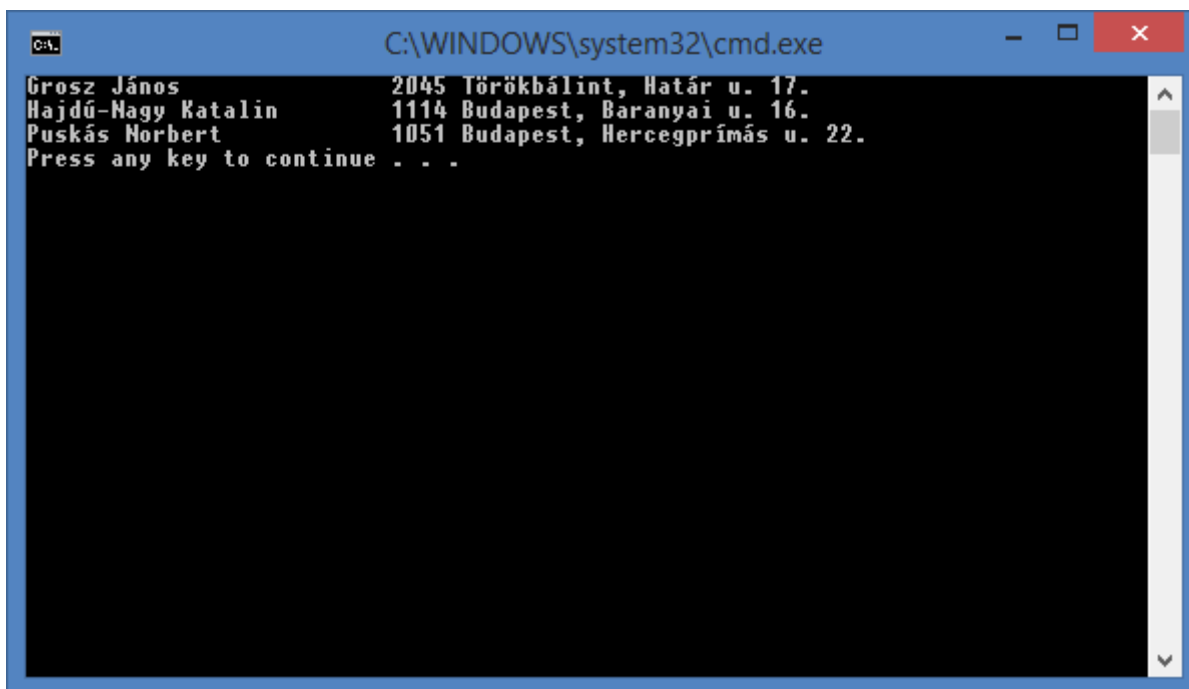
Készíts egy függvényt VevoTelephelyListazas() néven mely a vevők neveit és központi telephelyeinek címait listázza ki. Az EDMX varázsló a KözpontiTelephelyhez tartozó tulajdonságokat nem tudja automatikusan jól elnevezni. Az egyszerűbb munka érdekében módosítsd az edmx szerkesztőben a KozpontiTelephely tulajdonság nevét KozpontiTelephelyID-re, majd a Telephely1 navigációs tulajdonságot KozpontiTelephelyre.

#### Megoldás:

```
static void VevoTelephelyListazas()
{
    using (var context = new LABOREntities())
    {
        var query = from v in context.Vevo
                    orderby v.Nev
                    select new
                    {
                        Nev = v.Nev,
                        Telep = v.KozpontiTelephely.IR + " " +
                            v.KozpontiTelephely.Varos + ", " +
                            v.KozpontiTelephely.Utca
                    };

        foreach (var vevo in query)
```

```
        {  
            Console.WriteLine("{0, -25} {1, 15}", vevo.Nev, vevo.Telep);  
        }  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
Grosz János      2045 Törökbálint, Határ u. 17.  
Hajdú-Nagy Katalin 1114 Budapest, Baranyai u. 16.  
Puskás Norbert   1051 Budapest, Hercegprímás u. 22.  
Press any key to continue . . .
```

#### Indoklás:

Korábbi feladatban is láttunk anonim osztályt használni, aminek név és cím tulajdonsága van.

## 12.FELADAT

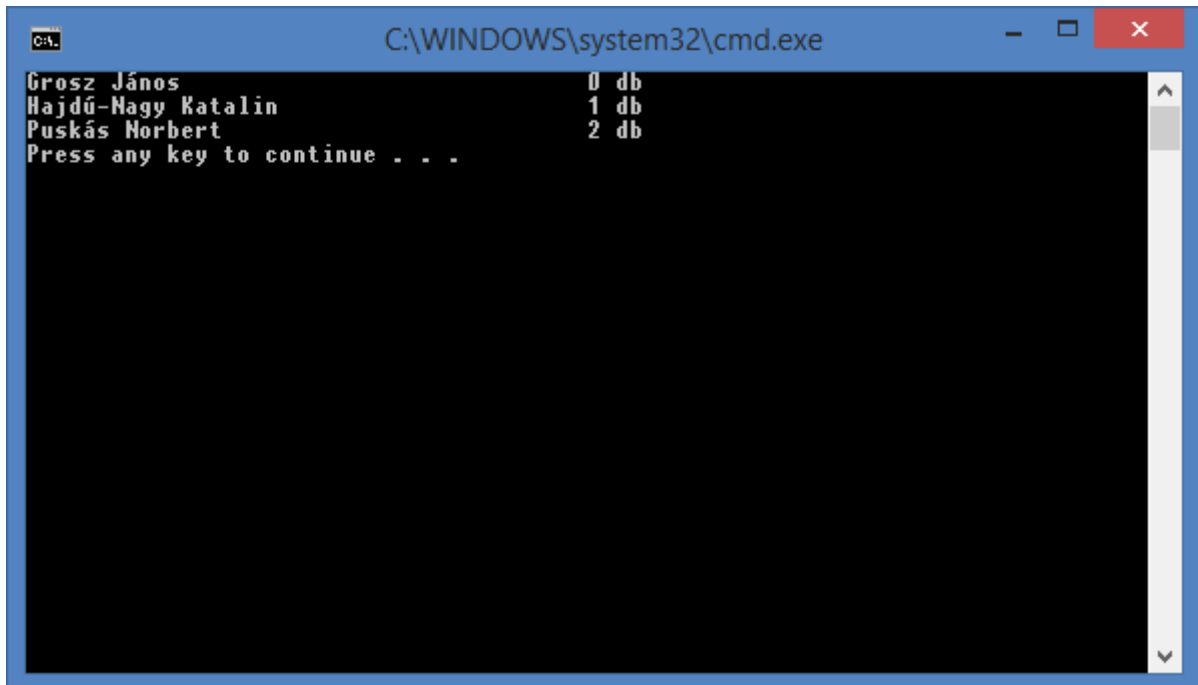
### Feladat neve és leírása:

Készíts egy függvényt `VevoMegrendelesek()` néven, mely felsorolja a vevőket és minden vevő mellett feltünteti az eddigi, központi telephelyről feladott megrendeléseinek számát! Ügyelj rá, hogy a listázás egyetlen SQL utasítás végrehajtásával történjen.

### Megoldás:

```
static void VevoMegrendelesek()  
{  
    using (var context = new LABOREntities())  
    {  
        var query = from v in context.Vevo  
                     orderby v.Nev  
                     select new  
                     {  
                         Nev = v.Nev,  
                         Ossz = v.KozpontiTelephely.Megrendeles.Count()  
                     };  
  
        foreach (var vevo in query)
```

```
        {  
            Console.WriteLine("{0, -25} {1, 15} db", vevo.Nev, vevo.Ossz);  
        }  
    }  
}
```



```
C:\WINDOWS\system32\cmd.exe  
Grosz János          0 db  
Hajdú-Nagy Katalin  1 db  
Puskás Norbert      2 db  
Press any key to continue . . .
```

#### Indoklás:

Az osztályban létrehozok egy attribútumot, amiben megszámolom a megrendeléseket.

#### 14.FELADAT

##### Feladat neve és leírása:

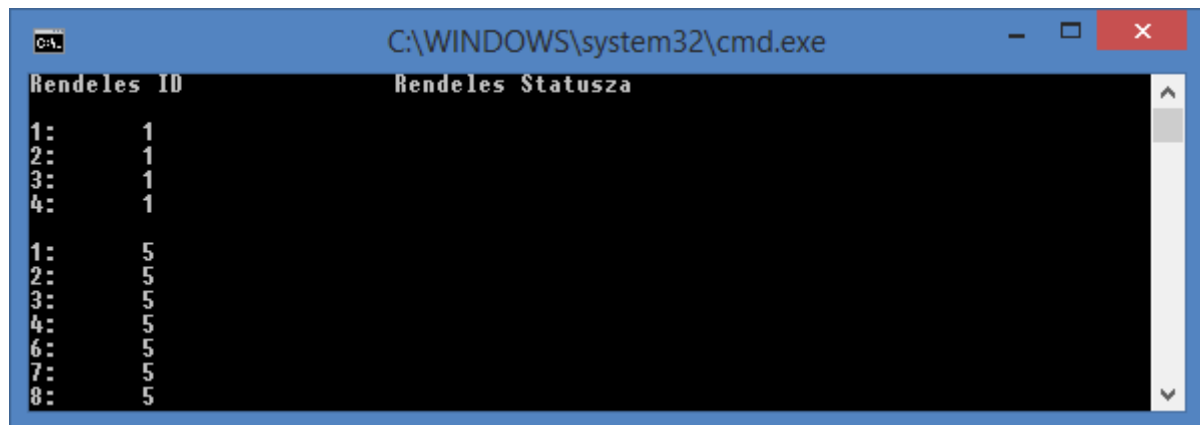
Készíts egy függvényt `Kiszallitas()` néven, mely minden megrendelés státuszát 5-re („Kiszállítva”) állítja. A módosítás során a megrendelés tételeinek a státuszát is módosítani kell.

##### Megoldás:

```
static void Kiszallitas()  
{  
    using (var context = new LABOREntities())  
    {  
        var query = from m in context.Megrendeles  
                     where m.StatuszID != 5  
                     select m;  
  
        foreach (var rendeles in query)  
        {  
            Console.WriteLine("{0}:\t{1}", rendeles.ID, rendeles.StatuszID);  
            rendeles.StatuszID = 5;  
            foreach (var tetel in rendeles.MegrendelesTetel)  
            {
```

```
        tete1.StatuszID = 5;
    }
}

context.SaveChanges();
}
```



Rendeles ID	Rendeles Statusza
1:	1
2:	1
3:	1
4:	1
1:	5
2:	5
3:	5
4:	5
6:	5
7:	5
8:	5

#### Indoklás:

A Kiszállítva státusz ID-ja 5-ös. Lekérdeztem minden olyan rekordot, ami még nincs kiszállítva és átállítom az azonosítót 5-re. Kiírom az elején a megrendelés ID-t és a státusz ID-t, majd miután lefutott a metódus, akkor az összes kiszállított megrendelést is.

VÉLEMÉNY

1. **Milyennek értékeled a gyakorlat nehézségét?**
2. **Mennyire érzed hasznosnak a gyakorlat anyagát?**
3. **Milyen javaslataid lennének a gyakorlat felépítésével kapcsolatban?**