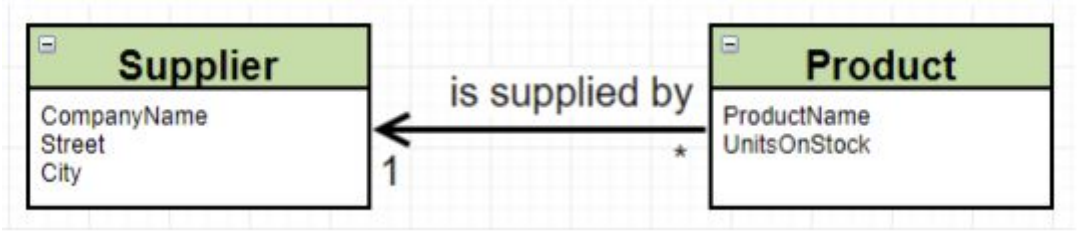
Bazy danych – Entity

Imię i nazwisko: Błażej Kustra

Tydzień B, czwartek 12:50

I. Na zajęciach - podstawy oraz inicjacja samego projektu
II. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej



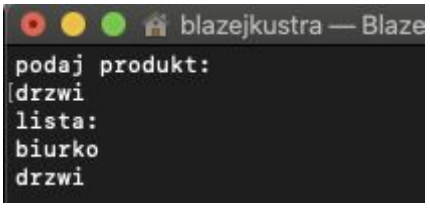
Stworzyłem klasę Supplier oraz zmodyfikowałem klasę Product, ProductContext.

```
1   using System;
2   using System.ComponentModel.DataAnnotations;
3   using System.Text;
4   using System.Collections.Generic;
5
6   namespace BlazejKustraHellow
7   {
8       public class Supplier
9       {
10          [Key]
11          public int SupplierId { get; set; }
12          public string CompanyName { get; set; }
13          public string Street { get; set; }
14          public string City { get; set; }
15      }
16  }
```

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace BlazejKustraHellow
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsInStock { get; set; }

        [ForeignKey("Supplier")]
        public int SupplierId { get; set; }
        public Supplier supplier { get; set; }
    }
}
```

```
ng System;
ng Microsoft.EntityFrameworkCore;

espace BlazejKustraHellow

 public class ProductContext : DbContext
 {
     public DbSet<Product> Products { get; set; }
     protected override void OnConfiguring(DbContextOptionsBuilder options
         => options.UseSqlite("DataSource=Product.db");

     public DbSet<Supplier> Suppliers { get; set; }
 }
```

A. Dodaj do bazy nowy Produkt



B. Stworz nowego dostawce.
C. Znajdz poprzednio wprowadzony produkt i ustaw jego dostawce na właśnie dodanego.

main.Products [Product.db] | main.Suppliers [Product.db]

1 row | Tx: Auto | DDL | Tab-se...d (T

<Filter criteria>

	SupplierID	CompanyName	Street	City
1	1	Drewit	none	none

main.Products [Product.db]

2 rows | Tx: Auto | DDL | Tab-se...d (T

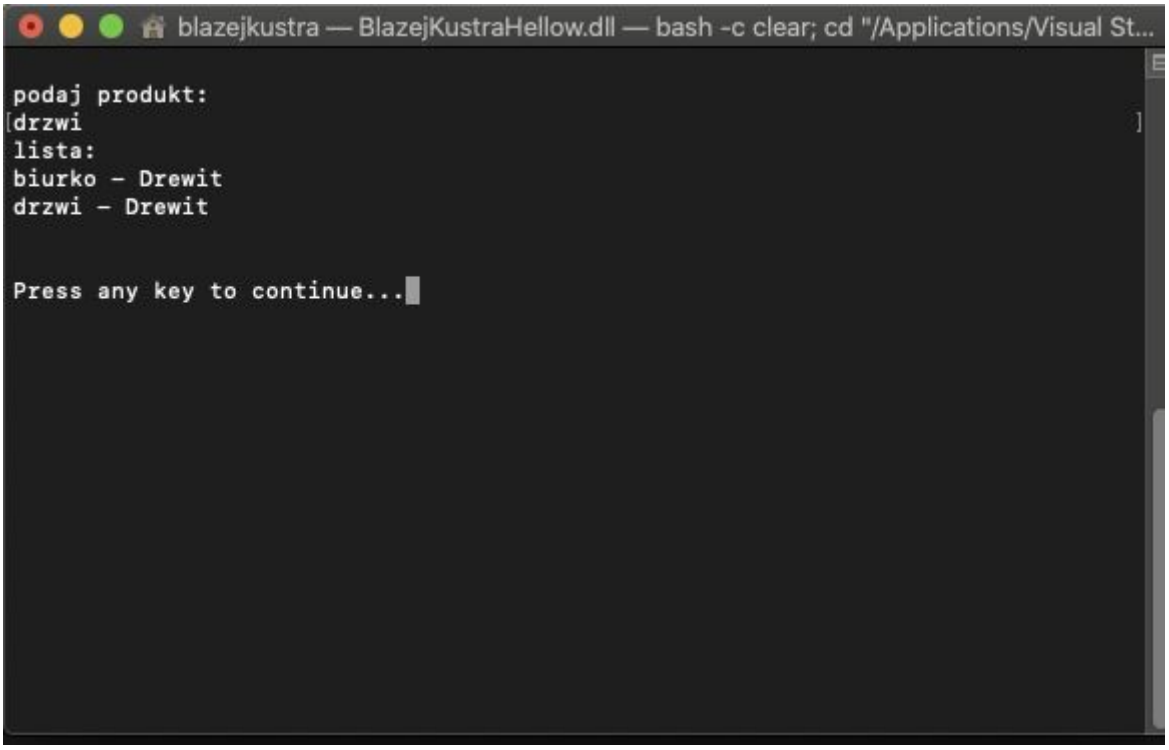
<Filter criteria>

	ProductID	ProductName	UnitsInStock	SupplierID
1	1	biurko	0	1
2	2	drzwi	0	1

D. Wyświetl wszystkie produkty wraz z nazwą dostawcy

E. Udokumentuj wykonane kroki oraz uzyskany rezultat (ogi wywołań sqlowych,describe table/diagram z datagrip, select * from….)

```
using System;
using System.Linq;
namespace BlazejKustraHellow
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine();
            Console.WriteLine("podaj produkt:");

            string productName = Console.ReadLine();
            Product product = new Product { ProductName = productName,
                                            UnitsInStock= 0 };
            ProductContext productContext = new ProductContext();

            Supplier supplier = new Supplier { CompanyName = "Drewit" };
            productContext.Suppliers.Add(supplier);
            productContext.SaveChanges();

            var supplier1 = (from prod in productContext.Suppliers
                       select prod).First();

            product.SupplierID = supplier1.SupplierID;
            productContext.Products.Add(product);
            productContext.SaveChanges();

            Console.WriteLine("lista:");
            var query = from p in productContext.Products
                        select p.ProductName;

            foreach( var item in query)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();
        }
    }
}

```



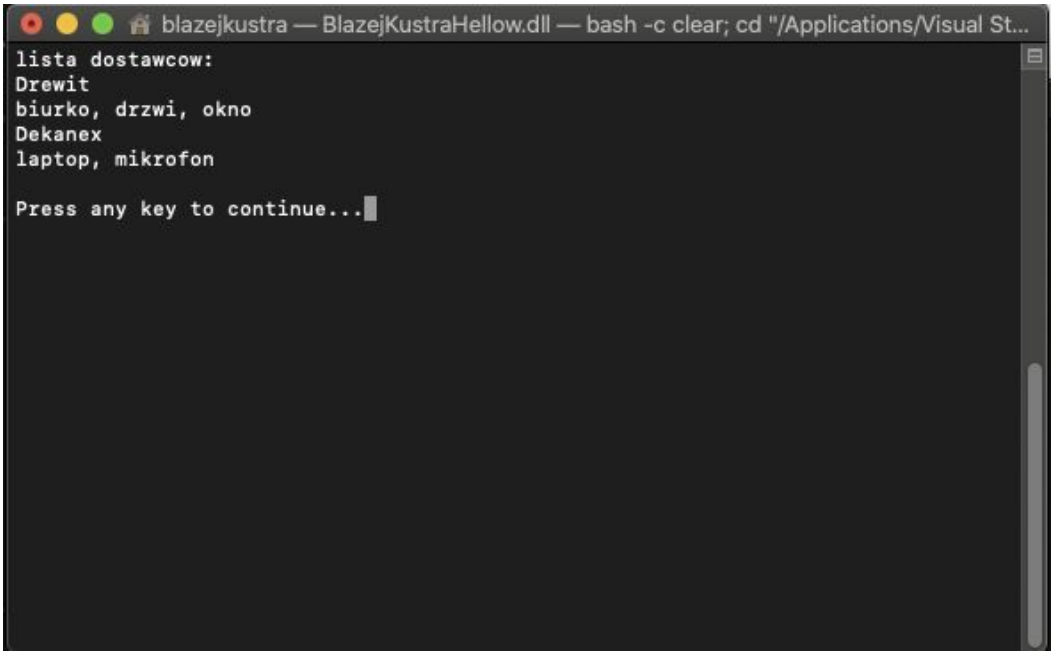
III. Odwróć relacje zgodnie z poniższym schematem

A. Stworz kilka produktow
B. Dodaj je do produktow dostarczanych przez nowo stworzonego dostawcę
C. Udokumentuj wykonane kroki oraz uzyskane rezultaty w obu wariantach (logi wywołań sqlowych, describe table/diagram z datagrip, select * from….)

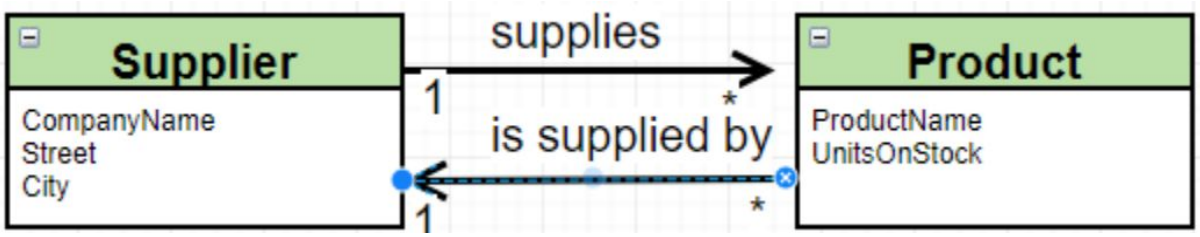
```cs
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace BlazejKustraHellow
{
    public class Product
    {
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsInStock { get; set; }
    }
}
```

```cs
using System;
using System.ComponentModel.DataAnnotations;
using System.Text;
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace BlazejKustraHellow
{
    public class Supplier
    {
        [Key]
        public int SupplierID { get; set; }
        public string CompanyName { get; set; }
        public string Street { get; set; }
        public string City { get; set; }
        public ICollection<Product> Products { get; set; }

        public Supplier()
        {
            Products = new Collection<Product>();
        }
    }
}
```

Dodałem nowego dostawcę i parę produktów a potem wypisałem wszystkie produkty odpowiednich dostawców:



IV. Zamodeluj relacje dwustronną jak poniżej:

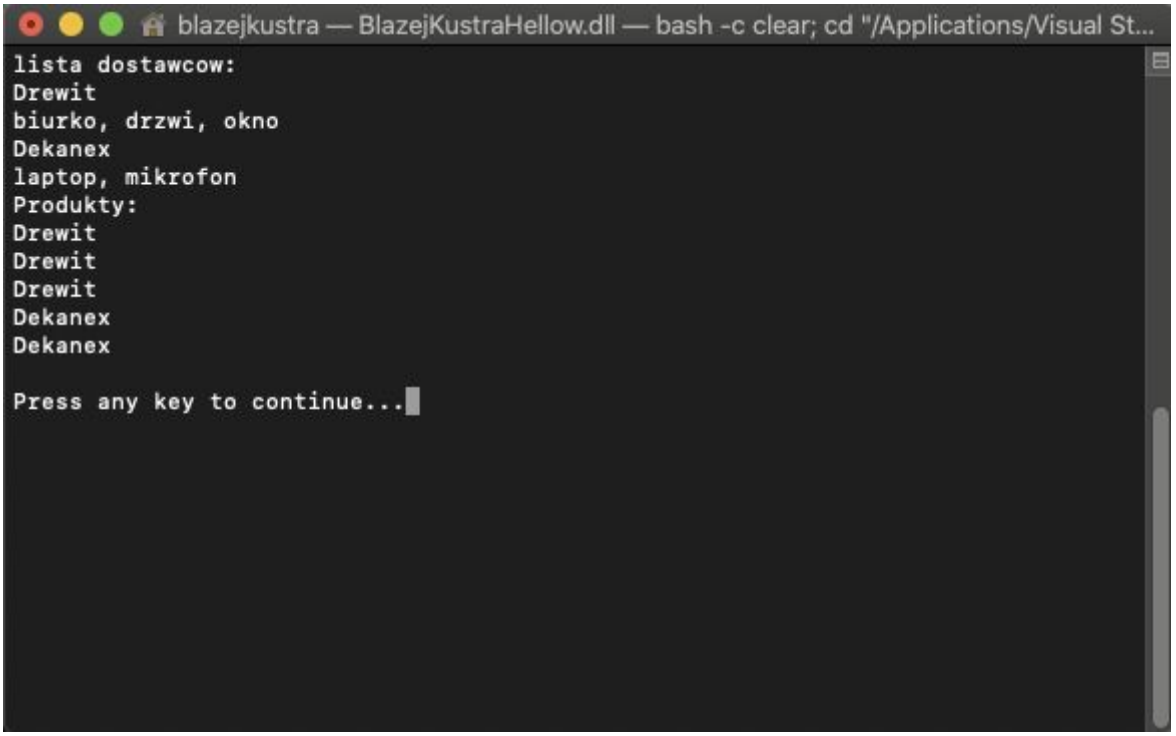


A. Tradycyjnie: Stworz kilka produktow
B. Dodaj je do produktow dostarczanych przez nowo stworzonego dostawcę (pamiętaj o poprawnej obsłudze dwustronności relacji)
C. Udokumentuj wykonane kroki oraz uzyskane rezultaty (logi wywołań sqlowych, describe table/diagram z datagrip, select * from….)

Do klasy Product dodałem dostawcę tak zeby mozna bylo do niego sie odwolywac.

```
1   using System;
2   using System.ComponentModel.DataAnnotations.Schema;
3
4   namespace BlazejKustraHellow
5   {
6       public class Product
7       {
8           public Supplier Supplier { get; set; }
9           public int ProductID { get; set; }
10          public string ProductName { get; set; }
11          public int UnitsInStock { get; set; }
12      }
13  }
```

Wypisałem jak poprzednio produkty dla dostawców, oraz dla każdego produktu jego dostawcę:



```
using System;
using System.Linq;
namespace BlazejKustraHellow
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine();
            Console.WriteLine("podaj produkt:");

            string productName = Console.ReadLine();
            Product product = new Product
            {
                ProductName = productName,
                UnitsInStock = 0
            };
            ProductContext productContext = new ProductContext();

            Supplier supplier = new Supplier { CompanyName = "dekanex" };
            productContext.Suppliers.Add(supplier);
            productContext.SaveChanges();

            var supplier1 = (from prod in productContext.Suppliers
                             select prod).First();

            product.Supplier= supplier1;
            productContext.Products.Add(product);
            productContext.SaveChanges();

            Console.WriteLine("lista:");
            var query = from p in productContext.Products
                        select p;

            foreach (var item in query)
            {
                Console.Write(item.Supplier);
            }
            Console.WriteLine();
        }
    }
}
```

V. Dodaj klase Category z property int CategoryID, String Name oraz listą produktów

```
using System;
using System.Collections.Generic;

namespace BlazejKustraHellow
{
    public class Category
    {
        public int CategoryID { get; set; }
        public string CategoryName { get; set; }
        public List<Product> Products { get; set; }
    }
}
```

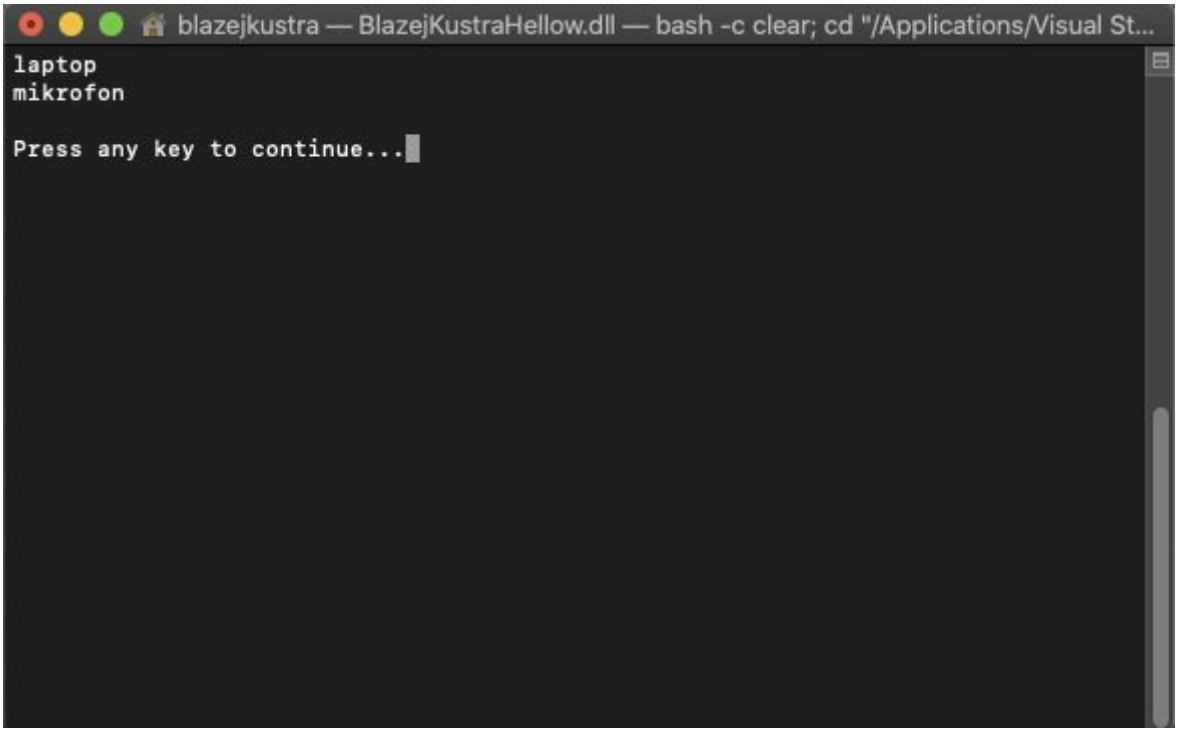
A. Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.

```
using System;
using System.ComponentModel.DataAnnotations.Schema;

namespace BlazejKustraHellow
{
    public class Product
    {
        public Supplier Supplier { get; set; }
        public Category Category { get; set; }
        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsInStock { get; set; }
    }
}
```

```
using System;
using Microsoft.EntityFrameworkCore;

namespace BlazejKustraHellow
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }
        public DbSet<Category> Categories { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder opti
        => options.UseSqlite("DataSource=Product.db");
    }
}
```

B. Stworz kilka produktow i kilka kategorii
C. Dodaj kilka produktów do wybranej kategorii
D. Wydobądź produkty z wybranej kategorii oraz kategorię do której należy wybrany produkt

```
var productsFromCatergory = productContext.Categories.Include(
    c => c.Products.Where(c => c.CategoryName == "agd"));

foreach(var c in productsFromCatergory)
{
  foreach(var p in c.Products)
  {
        Console.WriteLine(p.ProductName);
  }
}
```

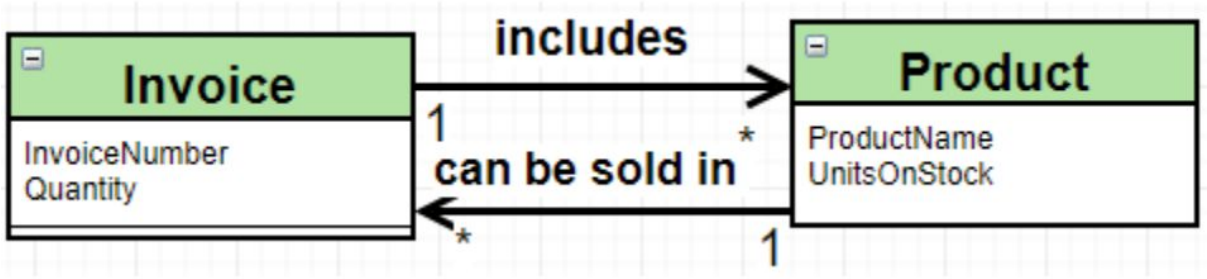


	CategoryID	CategoryName
1	1	meble
2	2	agd

	ProductID	ProductName	UnitsInStock	SupplierID	CategoryID
1	1	biurko	0	1	1
2	2	drzwi	0	1	1
3	3	okno	0	1	1
4	4	laptop	0	2	2
5	5	mikrofon	0	2	2

	SupplierID	CompanyName	Street	City
1	6	Drewit	none	none
2	7	dekanex	none	none

VI. Zamodeluj relacje wiele-do-wielu, jak poniżej:



A. Zmodyfikuj produkty dodając wskazanie na kategorie do której należy.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Collections.ObjectModel;
4
5   namespace BlazejKustraHellow
6   {
7       public class Category
8       {
9           public int CategoryID { get; set; }
10          public string CategoryName { get; set; }
11          public ICollection<Product> Products { get; set; }
12
13          public Category()
14          {
15              Products = new Collection<Product>();
16          }
17      }
18  }
```

```
1   using System;
2   namespace BlazejKustraHellow
3   {
4       public class InvoiceProduct
5       {
6           public int ProductID { get; set; }
7           public Product Product { get; set; }
8           public int InvoiceID { get; set; }
9           public Invoice Invoice { get; set; }
10      }
11  }
```

```cs
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace BlazejKustraHellow
{
    public class Invoice
    {
        public int InvoiceID { get; set; }
        public int InvoiceNumber { get; set; }
        public int Quantity { get; set; }
        public ICollection<InvoiceProduct> InvoiceProducts { get; set; }

        public Invoice()
        {
            InvoiceProducts = new Collection<InvoiceProduct>();
        }
    }
}
```

```cs
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel.DataAnnotations.Schema;

namespace BlazejKustraHellow
{
    public class Product
    {
        public ICollection<InvoiceProduct> InvoiceProducts { get; set; }

        public Supplier Supplier { get; set; }
        public Category Category { get; set; }

        public int ProductID { get; set; }
        public string ProductName { get; set; }
        public int UnitsInStock { get; set; }

        public Product()
        {
            InvoiceProducts = new Collection<InvoiceProduct>();
        }
    }
}
```

```
1   using System;
2   using Microsoft.EntityFrameworkCore;
3
4   namespace BlazejKustraHellow
5   {
6       public class ProductContext : DbContext
7       {
8           public DbSet<Product> Products { get; set; }
9           public DbSet<Supplier> Suppliers { get; set; }
10          public DbSet<Category> Categories { get; set; }
11
12          public DbSet<Invoice> Invoices { get; set; }
13          public DbSet<InvoiceProduct> InvoiceProducts { get; set; }
14
15          protected override void OnConfiguring(DbContextOptionsBuilder opti
16          => options.UseSqlite("DataSource=Product.db");
17
18          protected override void OnModelCreating(ModelBuilder model)
19          {
20              model.Entity<InvoiceProduct>().HasKey(item =>
21              new { item.ProductID, it  class BlazejKustraHellow.InvoiceProduct
22          }
23      }
24  }
```

	InvoiceID	InvoiceNumber	Quantity
1	1	1	4
2	2	2	3

	ProductID	InvoiceID
1	1	1
2	2	1
3	3	2
4	4	3
5	5	3

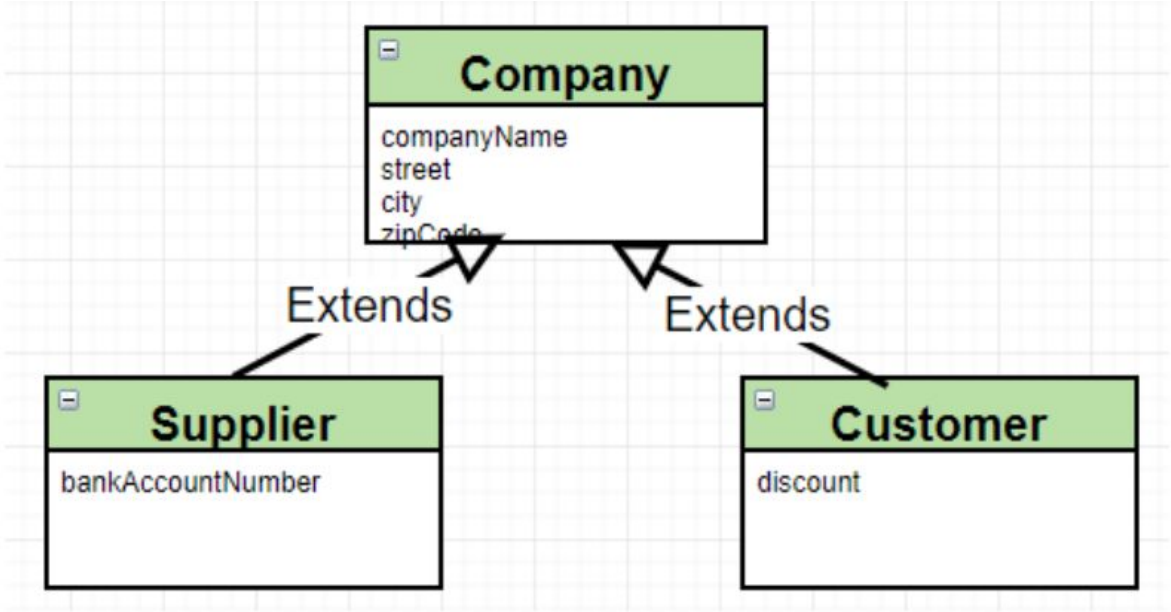
B. Pokaż produkty sprzedane w ramach wybranej faktury/transakcji

```
ProductContext productContext = new ProductContext();
var products = productContext.InvoiceProducts.Include(
    d => d.Product ).Where(d => d.InvoiceID == 3)
    .Select(d => d.Product.ProductName).ToList();

foreach(var prod in products)
{
    Console.WriteLine(prod)
}
```

C. Pokaż faktury w ramach których był sprzedany wybrany produkt

```
ProductContext productContext = new ProductContext();
var invoices = productContext.InvoiceProducts.Include(
    d => d.Invoice ).Where(p => p.ProductID == 3)
    .Select(d => d.Invoice.InvoiceNumber).ToList();

foreach(var inv in invoices)
{
    Console.WriteLine(inv);
}
```

VII. Dziedziczenie, Wprowadź do modelu następującą hierarchię:



A. Dodaj i pobierz z bazy kilka firm obu rodzajów stosując po kolei trzy różne strategie mapowania dziedziczenia. TablePerHierarchy, TablePerType, TablePerClass

```
1   using System;
2   namespace BlazejKustraHellow
3   {
4       public class Company
5       {
6           public int CompanyID { get; set; }
7           public string CompanyName { get; set; }
8           public string Street { get; set; }
9           public string City { get; set; }
10      }
11  }
```

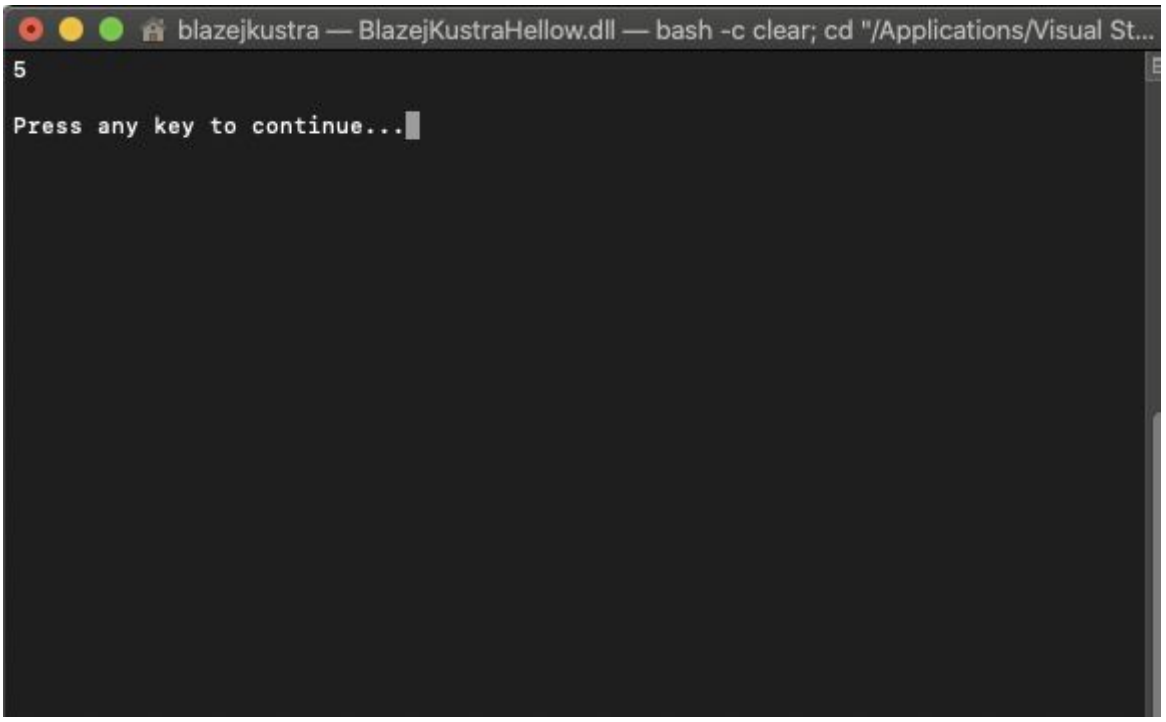
```
using System;
using System.ComponentModel.DataAnnotations;
using System.Text;
using System.Collections.Generic;
using System.Collections.ObjectModel;

namespace BlazejKustraHellow
{
    public class Supplier : Company
    {
        public int BankAcoountNumber { get; set; }
        public ICollection<Product> Products { get; set; }

        public Supplier()
        {
            Products = new Collection<Product>();
        }
    }
}
```

```
using System;
using Microsoft.EntityFrameworkCore;

namespace BlazejKustraHellow
{
    public class ProductContext : DbContext
    {
        public DbSet<Product> Products { get; set; }
        public DbSet<Supplier> Suppliers { get; set; }
        public DbSet<Category> Categories { get; set; }
        public DbSet<Company> Companies { get; set; }

        public DbSet<Invoice> Invoices { get; set; }
        public DbSet<InvoiceProduct> InvoiceProducts { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder opti
        => options.UseSqlite("DataSource=Product.db");

        protected override void OnModelCreating(ModelBuilder model)
        {
            model.Entity<InvoiceProduct>().HasKey(item =>
            new { item.ProductID, item.InvoiceID });

            model.Entity<Customer>();
            model.Entity<Supplier>();
        }
    }
}
```

```
1   using System;
2   using System.Linq;
3   using Microsoft.EntityFrameworkCore;
4
5   namespace BlazejKustraHellow
6   {
7       class Program
8       {
9           static void Main(string[] args)
10          {
11              Customer customer = new Customer
12              {
13                  City = "Krakow",
14                  CompanyName = "Blazej ",
15                  Street = "Mydlniki ",
16                  Discount = 5,
17
18
19              };
20              Supplier supplier = new Supplier
21              {
22                  City = "Krakow",
23                  CompanyName = "Drewit",
24                  Street = "Pradnicka 21",
25                  BankAccountNumber = 123456789
26              };
27
28              ProductContext productContext = new ProductContext();
29
30              productContext.Companies.Add(customer);
31              productContext.Companies.Add(supplier);
32
33              productContext.SaveChanges();
34          }
35      }
36  }
```

```
sqlite> select * from Companies;
1|Blazej|Mydlniki|Krakow|Customer|5|
2|Drewit|Pradnicka 21|Krakow|Supplier||123456789
sqlite>
```

```
var query = productContext.Companies
    .OfType<Customer>().ToList();

foreach(var item in query)
{
    Console.WriteLine(item.Discount);
}
```



Reszty zadań nie udało mi się zrobić, ciągłe błędy spowodowały, że zacząłem się zastanawiać czy na pewno jest to wykonalne, okazuje się, że od 3 wersji EF ToTable() straciło wsparcie, dlatego nie zrobiłem dwóch ostatnich podpunktów.

Old behavior

Before EF Core 3.0, `ToTable()` called on a derived type would be ignored since only inheritance mapping strategy was TPH where this isn't valid.

New behavior

Starting with EF Core 3.0 and in preparation for adding TPT and TPC support in a later release, `ToTable()` called on a derived type will now throw an exception to avoid an unexpected mapping change in the future.