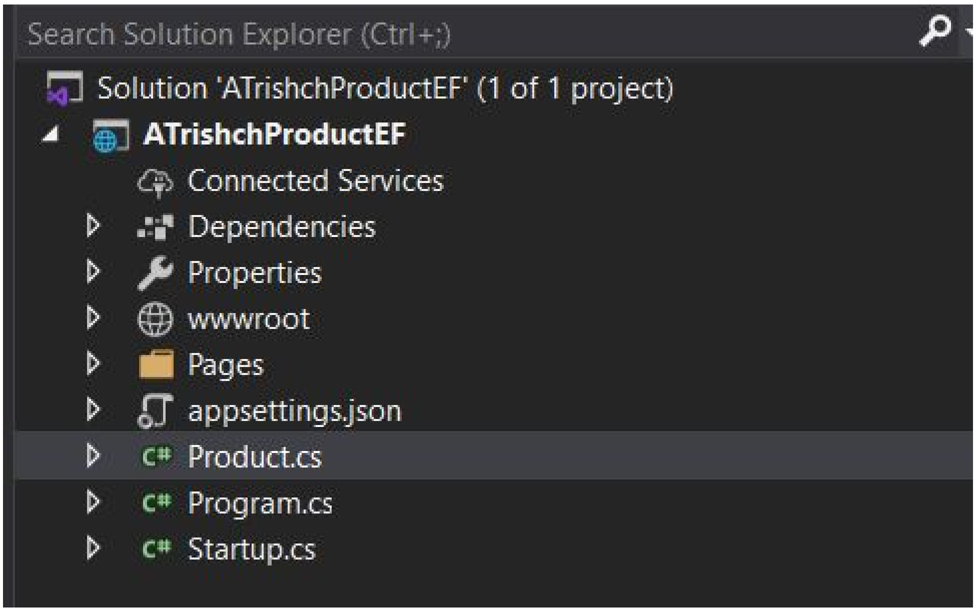
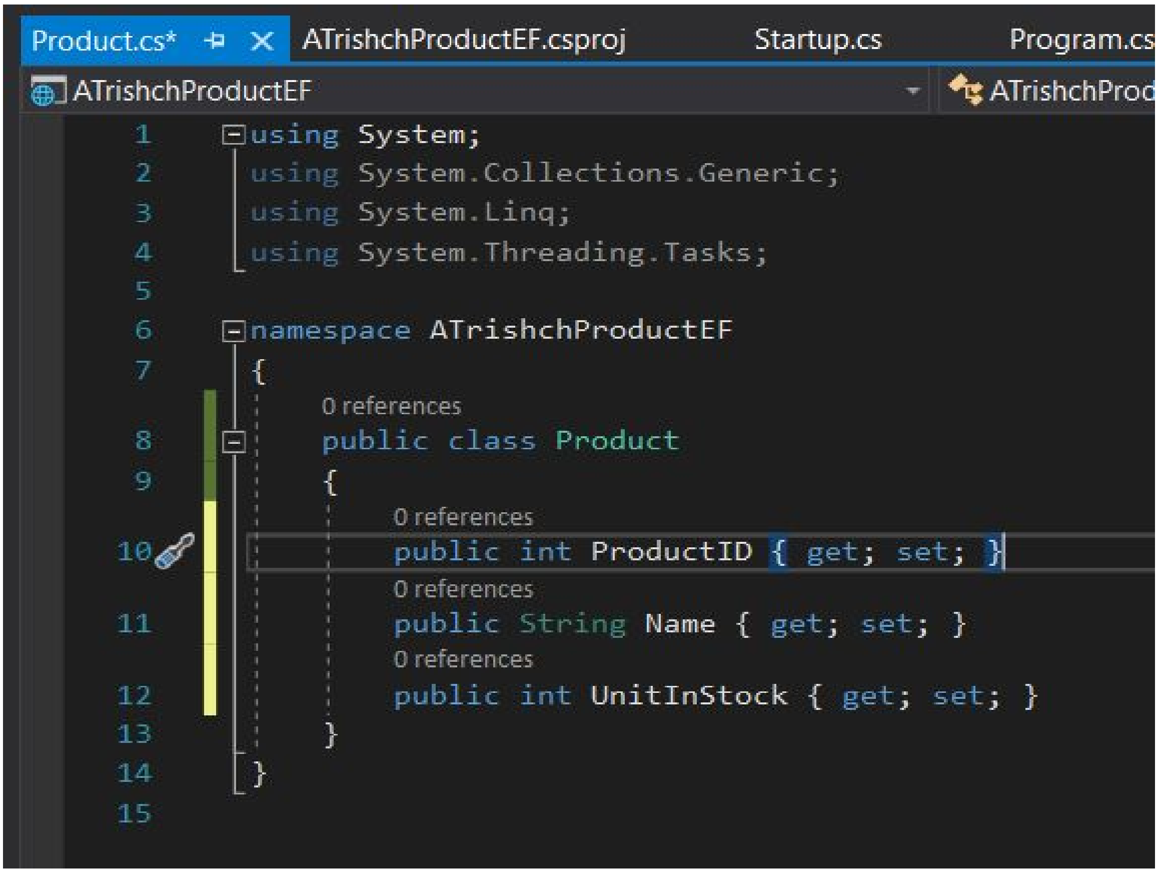
EntityFramework, LINQ2Entities – laboratorium

I. Code First

a. Stwórz projekt typu ConsoleApplication .Net Core. Nazwij go INazwiskoProdutcEF



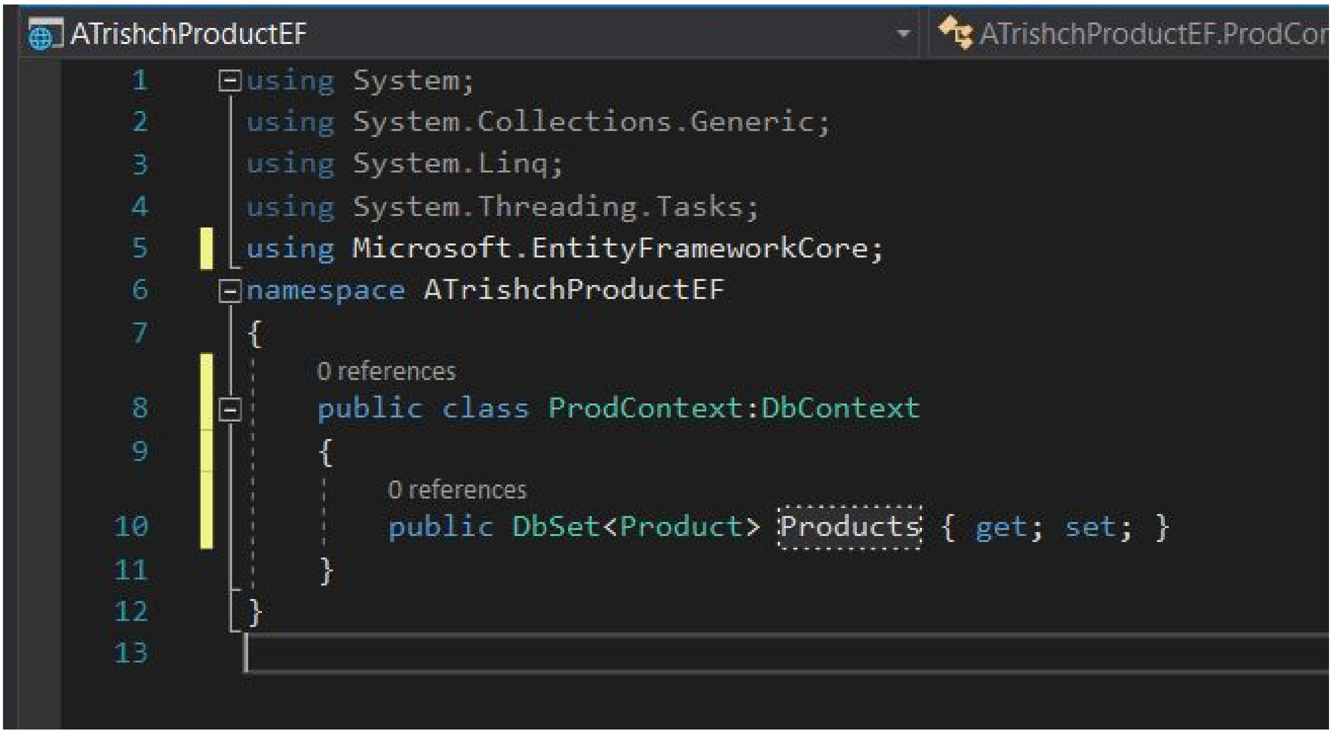
b. Dodaj klasę Product z polami int ProductID, string Name, int UnitsInStock. (Dodając do klasy property napisz prop I naciśnij dwa razy tabulator)



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace ATrishchProductEF
{
    0 references
    public class Product
    {
        0 references
        public int ProductID { get; set; }
        0 references
        public String Name { get; set; }
        0 references
        public int UnitInStock { get; set; }
    }
}
```

c. Stwórz klasę ProdContext dziedziczącą po DbContext.

d. Dodaj do klasy kontekstowej zbiór (DbSet) produktów i nazwij go Products



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
namespace ATrishchProductEF
{
    0 references
    public class ProdContext:DbContext
    {
        0 references
        public DbSet<Product> Products { get; set; }
    }
}
```

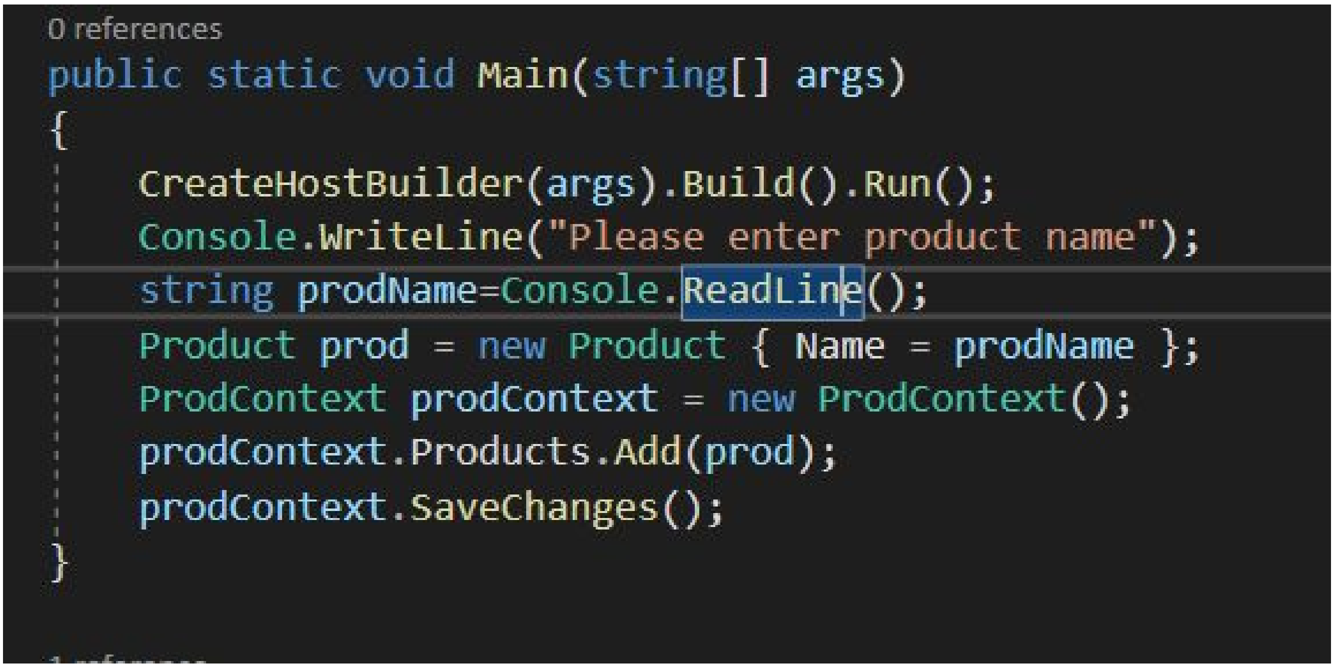
e. W Mainie (plik Program.cs)

i. poproś użytkownika o podanie nazwy produktu i zczytaj podana przez użytkownika nazwę ii. zainstancjonuj obiekt produktu ustawiając mu nazwę na tą zczytaną od użytkownika: iii. Stwórz instancje ProdContext'u

iv. dodaj zainstancjonowany obiekt do kontekstowej kolekcji Produktów

v. zapisz zmiany na kontekście

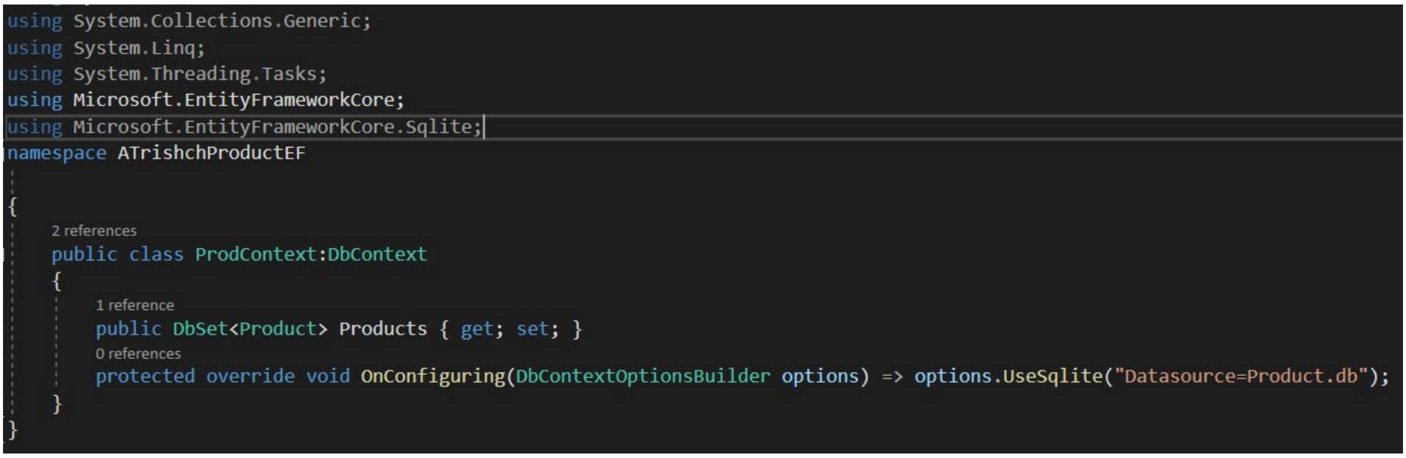
vi. Zbuduj i uruchom aplikacje vii.



```
0 references
public static void Main(string[] args)
{
    CreateHostBuilder(args).Build().Run();
    Console.WriteLine("Please enter product name");
    string prodName=Console.ReadLine();
    Product prod = new Product { Name = prodName };
    ProdContext prodContext = new ProdContext();
    prodContext.Products.Add(prod);
    prodContext.SaveChanges();
}
```

viii.Skonfigurujmy nasz kontekst, żeby wiedział do jakiej bazy chcemy się łączyć. Jednym ze sposobów jest nadpisanie w klasie naszego kontekstu metody OnConfiguring.

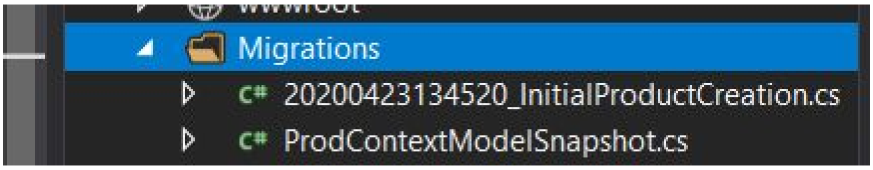
ix. Zbuduj i uruchom aplikacje.



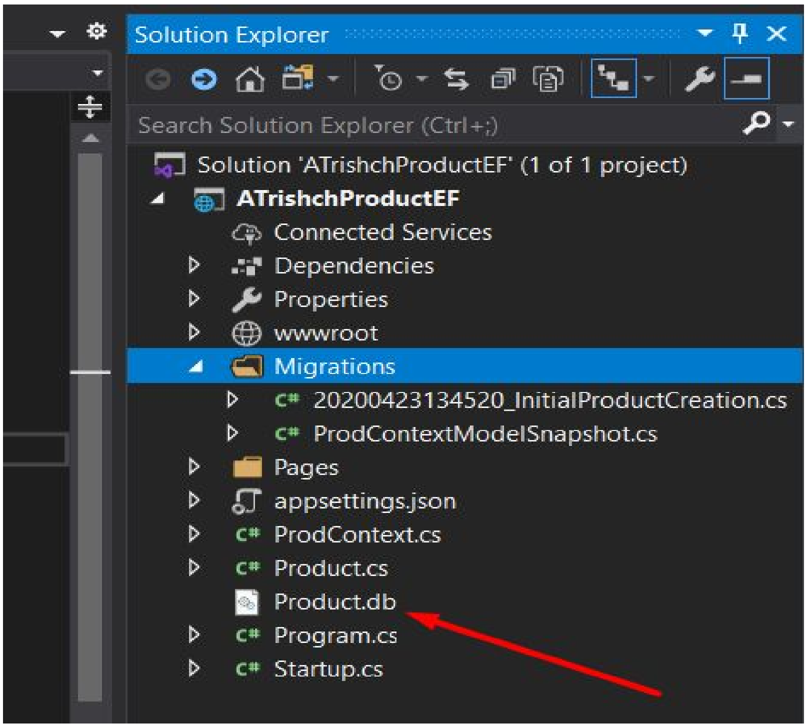
```
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Sqlite;
namespace ATrishchProductEF

{
    2 references
    public class ProdContext:DbContext
    {
        1 reference
        public DbSet<Product> Products { get; set; }
        0 references
        protected override void OnConfiguring(DbContextOptionsBuilder options) => options.UseSqlite("Datasource=Product.db");
    }
}
```

x. Dostaniesz wyjątek mówiący o tym, że nie istnieje tabela produktów. No i to prawda – bo generalnie w ogóle nie istnieje baza danych z którą chcemy pracować. Żeby to rozwiązać musimy wykonać dwa kroki

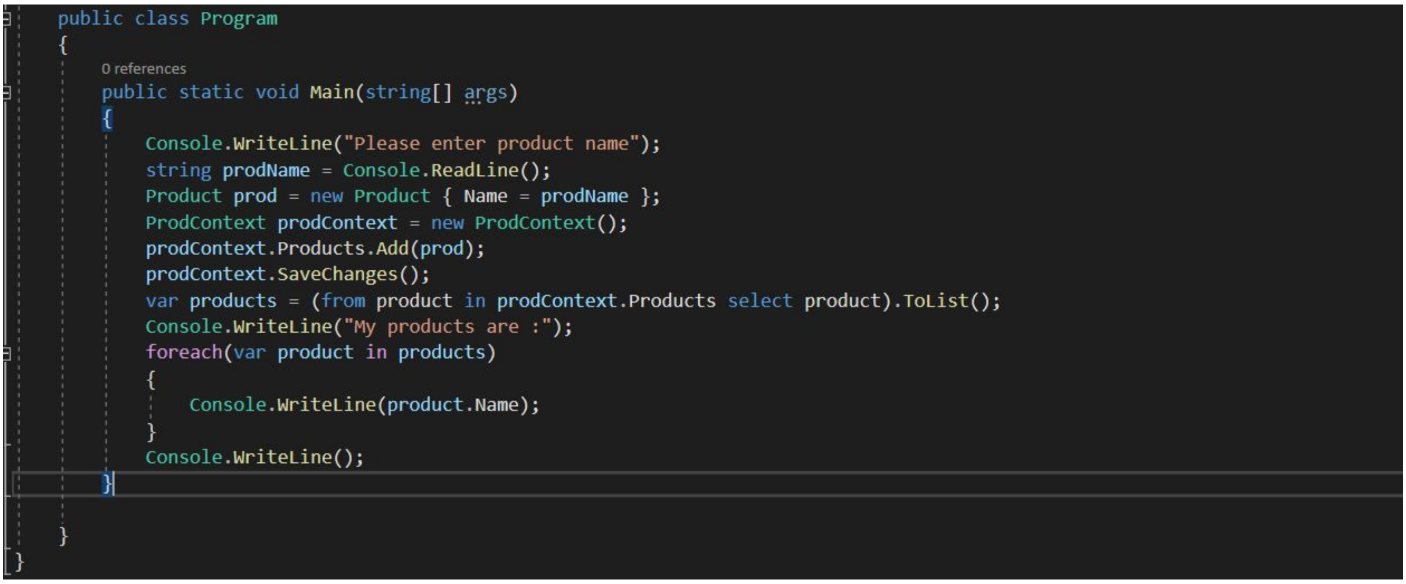
1. dotnet ef migrations add InitialProductCreation


wwwroot
Migrations
20200423134520_InitialProductCreation.cs
ProdContextModelSnapshot.cs

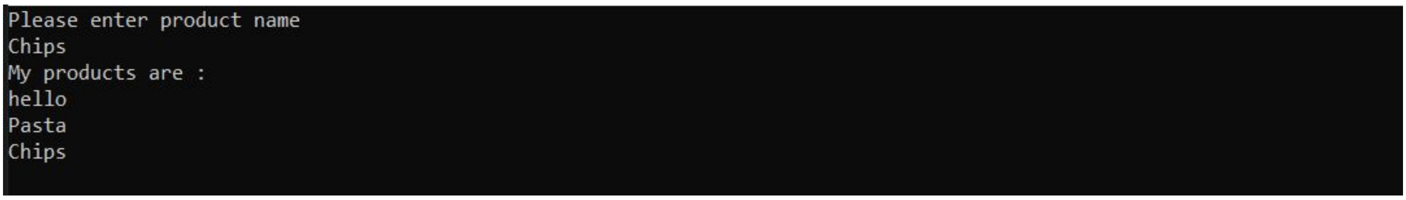
2. dotnet ef database update



3. Dopisz w mainie fragment kodu pobierający oraz wyświetlający dostępne Produkty. Jeżeli przy pisaniu zapytania o produkty zgłasza błąd typu „polecenie select jest nieznane” dodaj do usingów System.Linq



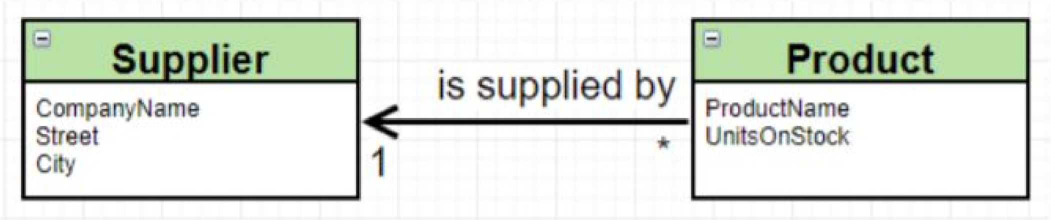
```
public class Program
{
    0 references
    public static void Main(string[] args)
    {
        Console.WriteLine("Please enter product name");
        string prodName = Console.ReadLine();
        Product prod = new Product { Name = prodName };
        ProdContext prodContext = new ProdContext();
        prodContext.Products.Add(prod);
        prodContext.SaveChanges();
        var products = (from product in prodContext.Products select product).ToList();
        Console.WriteLine("My products are :");
        foreach(var product in products)
        {
            Console.WriteLine(product.Name);
        }
        Console.WriteLine();
    }

}
}
```



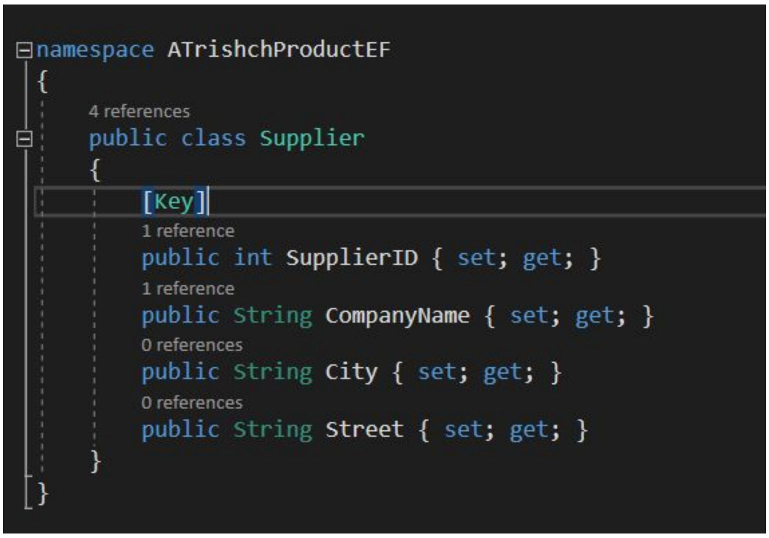
```
Please enter product name
Chips
My products are :
hello
Pasta
Chips
```

2.

II. Zmodyfikuj model wprowadzając pojęcie Dostawcy jak poniżej

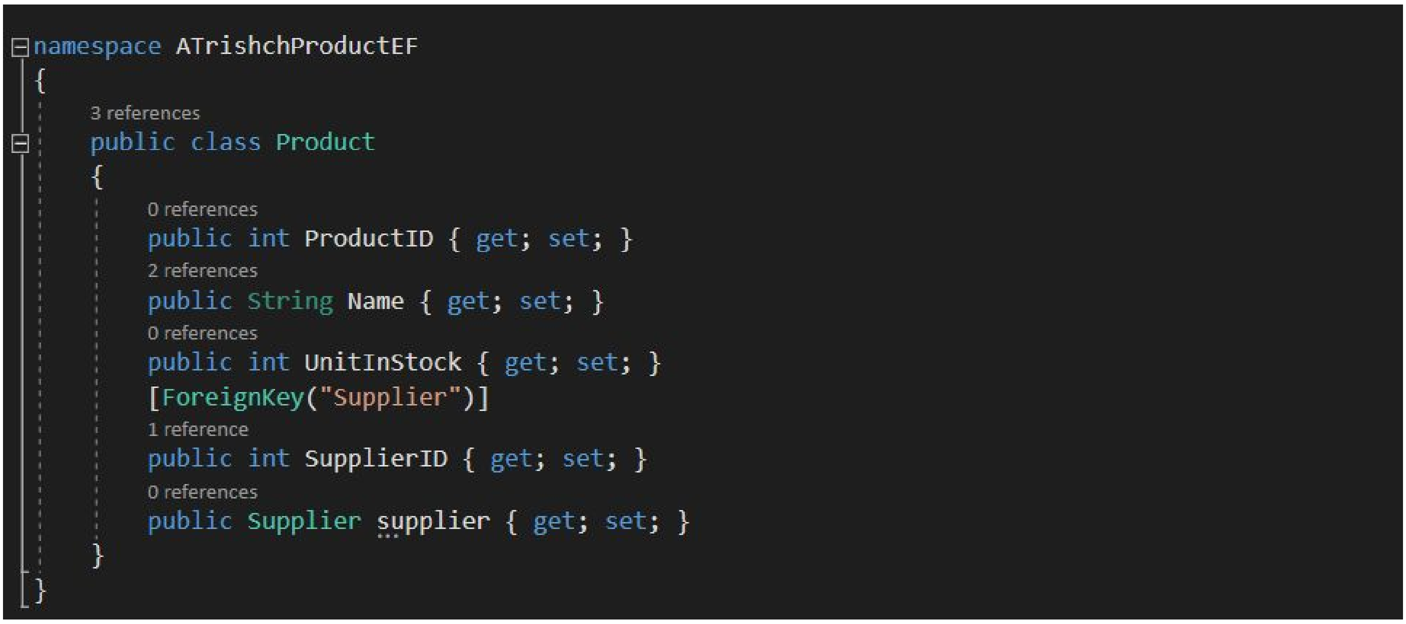


Została dodana klasa suppliers :



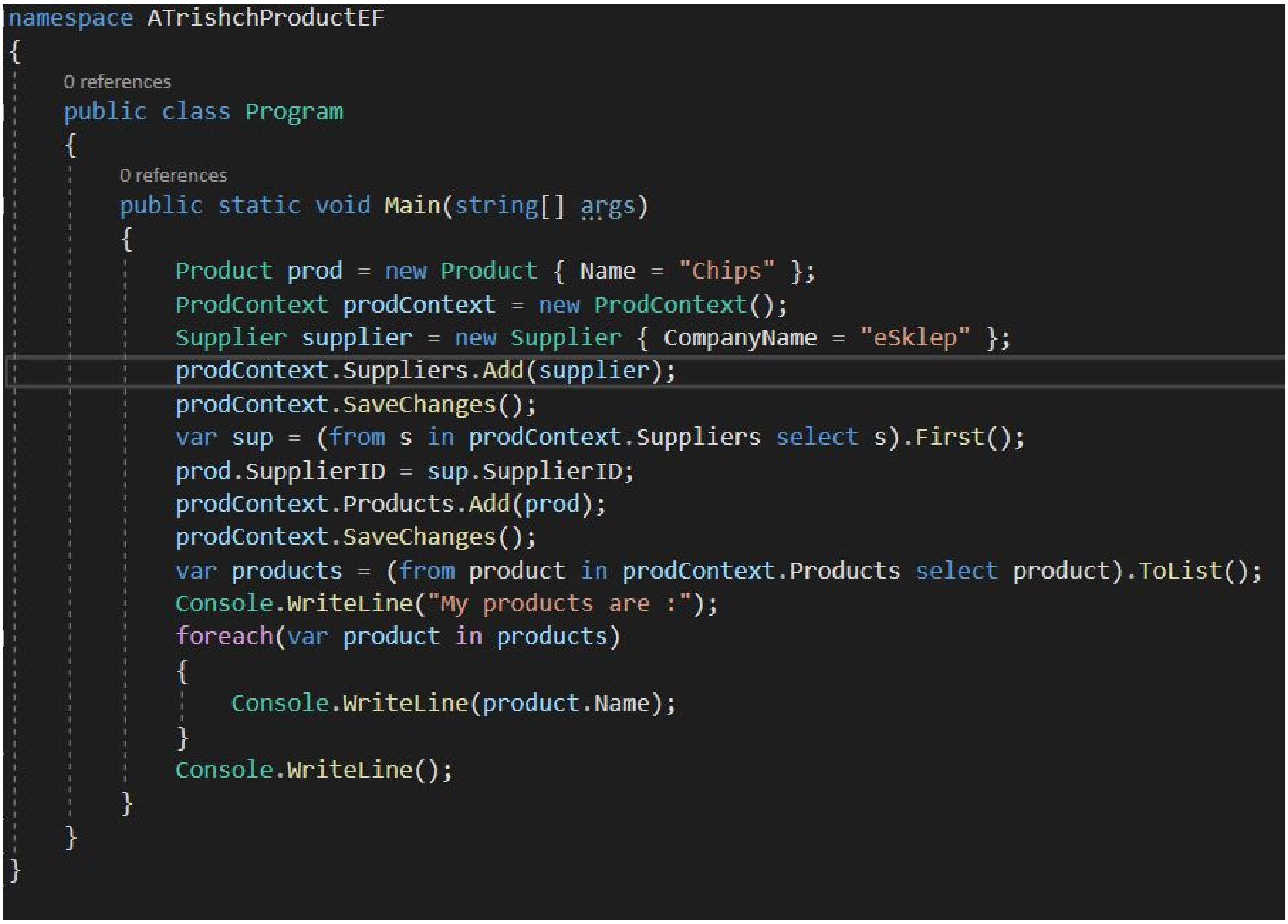
```
namespace ATrishchProductEF
{
    4 references
    public class Supplier
    {
        [Key]
        1 reference
        public int SupplierID { set; get; }
        1 reference
        public String CompanyName { set; get; }
        0 references
        public String City { set; get; }
        0 references
        public String Street { set; get; }
    }
}
```

Została zmodyfikowana klasa Product



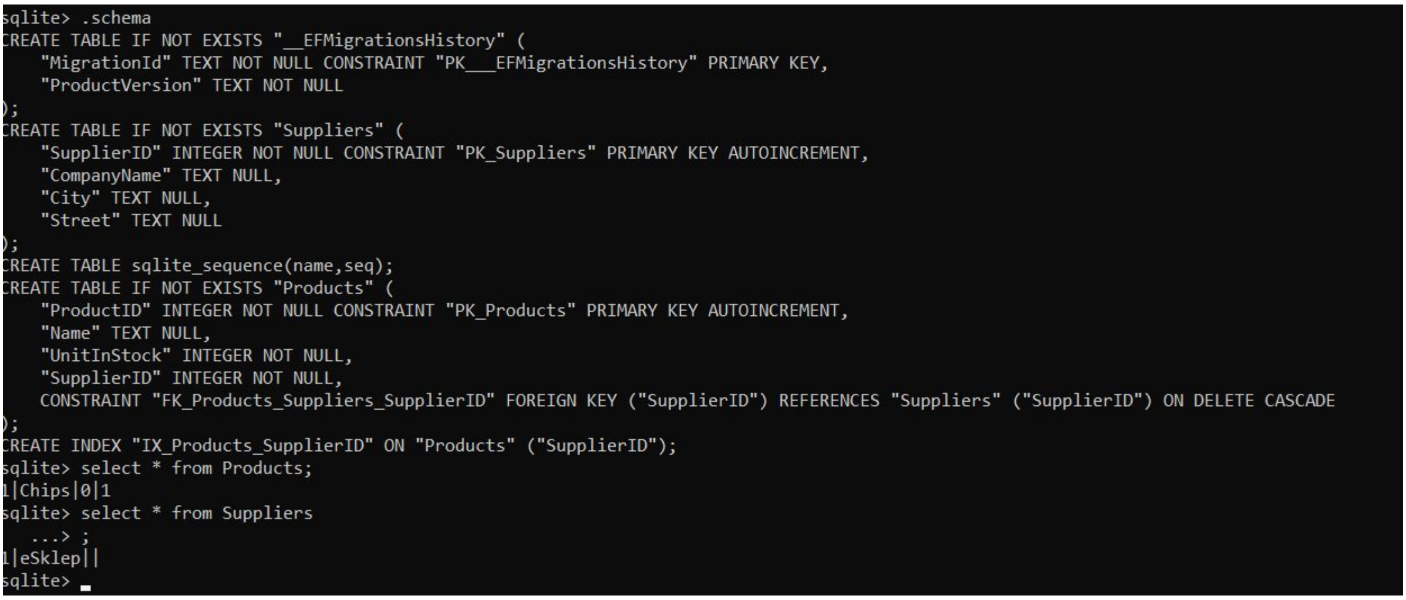
```
namespace ATrishchProductEF
{
    3 references
    public class Product
    {
        0 references
        public int ProductID { get; set; }
        2 references
        public String Name { get; set; }
        0 references
        public int UnitInStock { get; set; }
        [ForeignKey("Supplier")]
        1 reference
        public int SupplierID { get; set; }
        0 references
        public Supplier supplier { get; set; }
    }
}
```

Dodano dane dla sprawdzania poprawności działania.



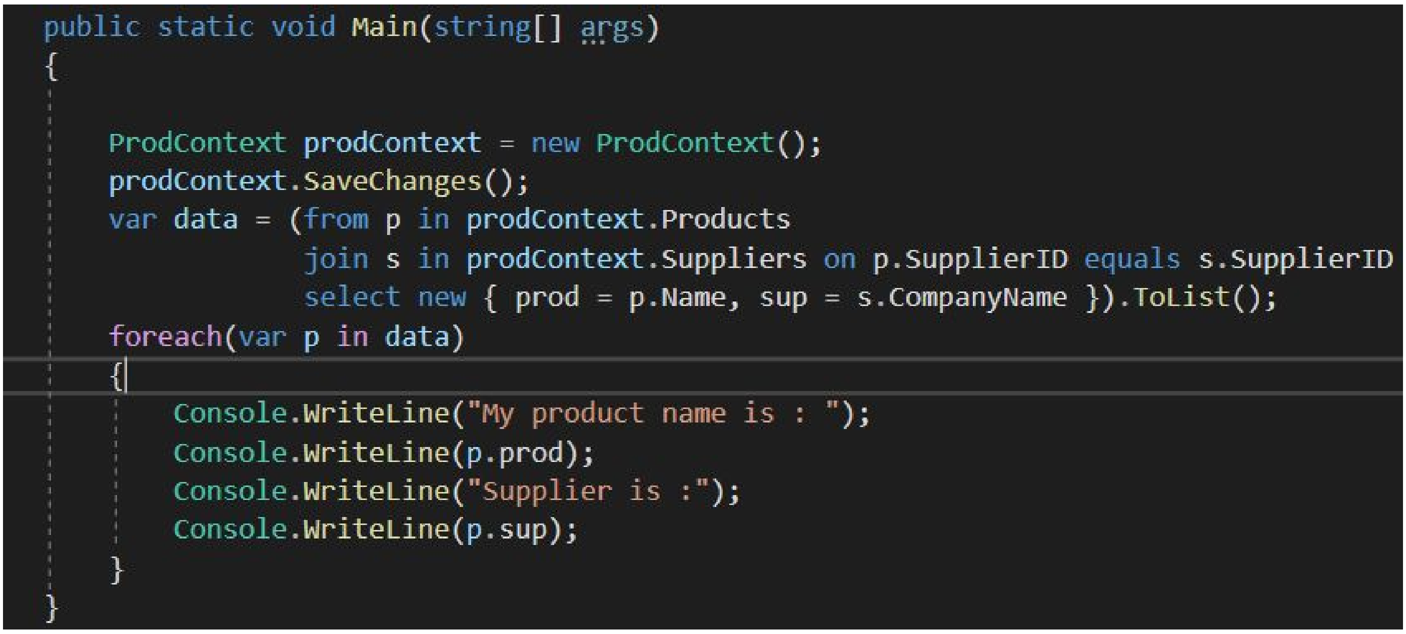
```
namespace ATrishchProductEF
{
    0 references
    public class Program
    {
        0 references
        public static void Main(string[] args)
        {
            Product prod = new Product { Name = "Chips" };
            ProdContext prodContext = new ProdContext();
            Supplier supplier = new Supplier { CompanyName = "eSklep" };
            prodContext.Suppliers.Add(supplier);
            prodContext.SaveChanges();
            var sup = (from s in prodContext.Suppliers select s).First();
            prod.SupplierID = sup.SupplierID;
            prodContext.Products.Add(prod);
            prodContext.SaveChanges();
            var products = (from product in prodContext.Products select product).ToList();
            Console.WriteLine("My products are :");
            foreach(var product in products)
            {
                Console.WriteLine(product.Name);
            }
            Console.WriteLine();
        }
    }
}
```

Wynik :

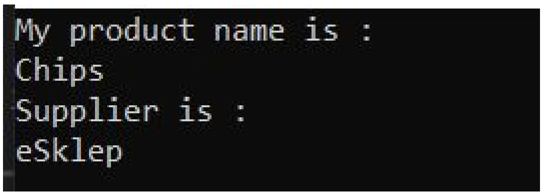


```
sqlite> .schema
CREATE TABLE IF NOT EXISTS "__EFMigrationsHistory" (
    "MigrationId" TEXT NOT NULL CONSTRAINT "PK___EFMigrationsHistory" PRIMARY KEY,
    "ProductVersion" TEXT NOT NULL
);
CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "City" TEXT NULL,
    "Street" TEXT NULL
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL,
    "UnitInStock" INTEGER NOT NULL,
    "SupplierID" INTEGER NOT NULL,
    CONSTRAINT "FK_Products_Suppliers_SupplierID" FOREIGN KEY ("SupplierID") REFERENCES "Suppliers" ("SupplierID") ON DELETE CASCADE
);
CREATE INDEX "IX_Products_SupplierID" ON "Products" ("SupplierID");
sqlite> select * from Products;
1|Chips|0|1
sqlite> select * from Suppliers
   ...> ;
1|eSklep||
sqlite>
```

Wyświetl wszystkie produkty wraz z nazwą dostawcy:



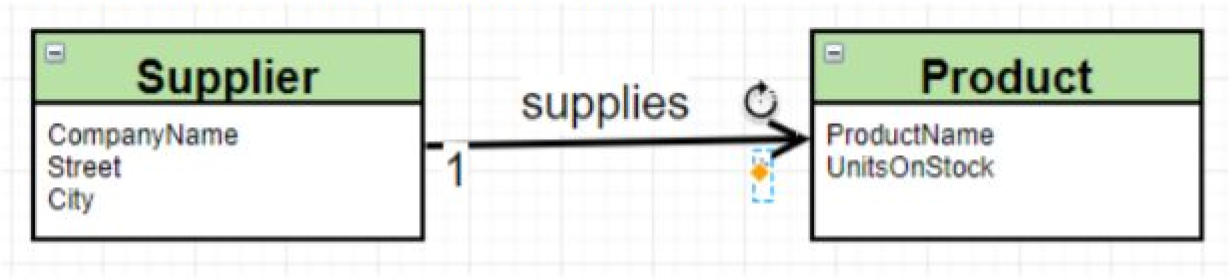
```
public static void Main(string[] args)
{

    ProdContext prodContext = new ProdContext();
    prodContext.SaveChanges();
    var data = (from p in prodContext.Products
                join s in prodContext.Suppliers on p.SupplierID equals s.SupplierID
                select new { prod = p.Name, sup = s.CompanyName }).ToList();
    foreach(var p in data)
    {
        Console.WriteLine("My product name is : ");
        Console.WriteLine(p.prod);
        Console.WriteLine("Supplier is :");
        Console.WriteLine(p.sup);
    }
}
```



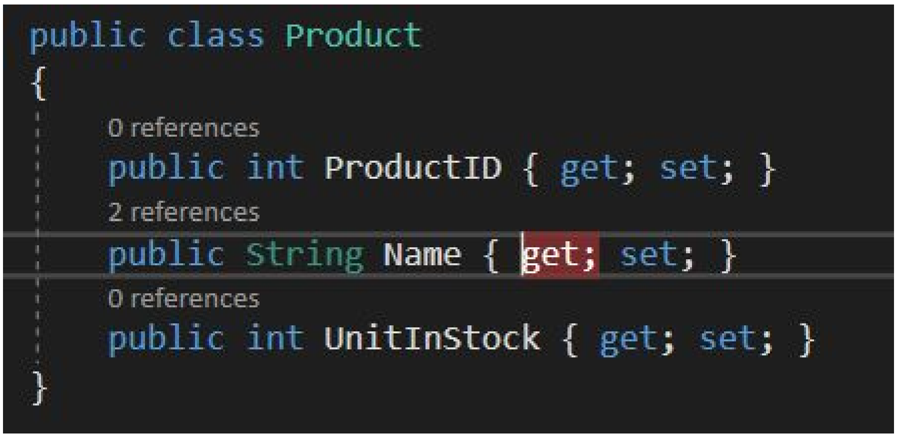
```
My product name is :
Chips
Supplier is :
eSklep
```

3.

III. Odwróć relacje zgodnie z poniższym schematem

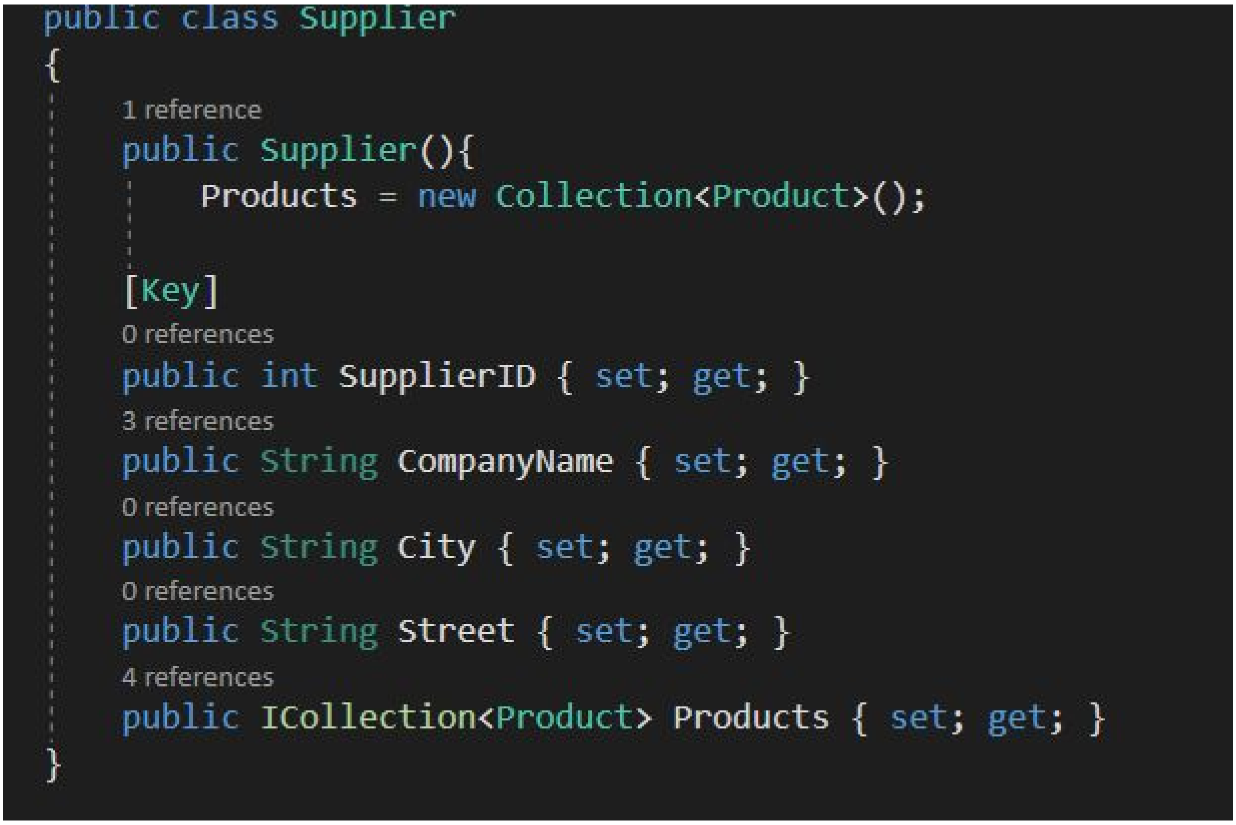


Zmieniłem klase Product



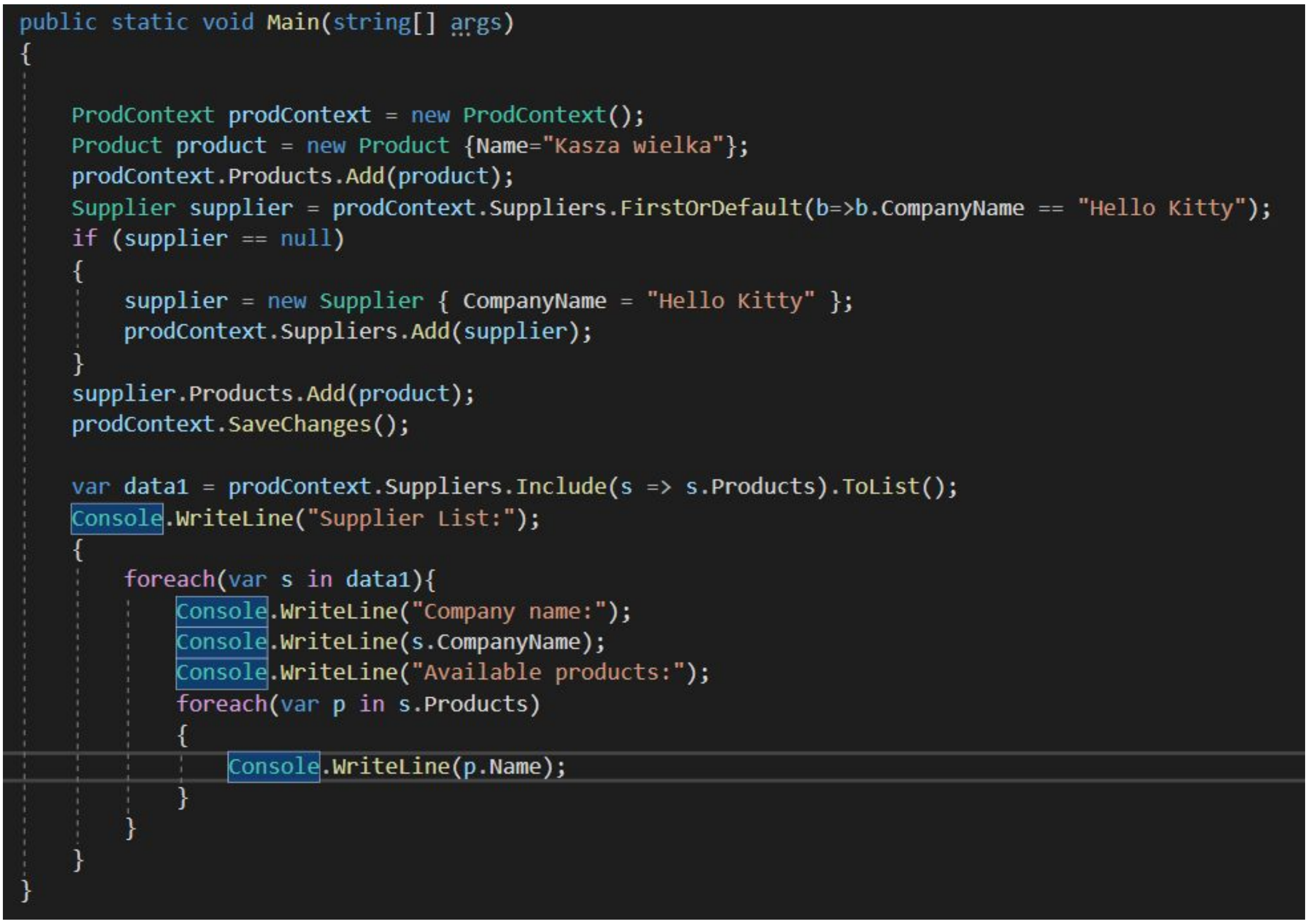
```
public class Product
{
    0 references
    public int ProductID { get; set; }
    2 references
    public String Name { get; set; }
    0 references
    public int UnitInStock { get; set; }
}
```

Dodałem w klase Supplier Kolekcje Products



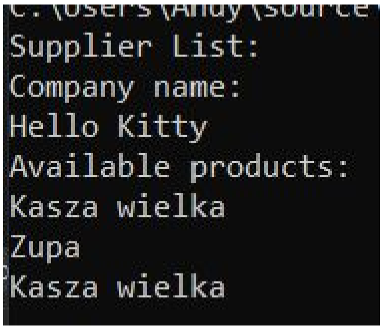
```
public class Supplier
{
    1 reference
    public Supplier(){
        Products = new Collection<Product>();

    [Key]
    0 references
    public int SupplierID { set; get; }
    3 references
    public String CompanyName { set; get; }
    0 references
    public String City { set; get; }
    0 references
    public String Street { set; get; }
    4 references
    public ICollection<Product> Products { set; get; }
}
```

Modifikacja Program.cs żeby sprawdzić poprawność działania programu



```
public static void Main(string[] args)
{

    ProdContext prodContext = new ProdContext();
    Product product = new Product {Name="Kasza wielka"};
    prodContext.Products.Add(product);
    Supplier supplier = prodContext.Suppliers.FirstOrDefault(b=>b.CompanyName == "Hello Kitty");
    if (supplier == null)
    {
        supplier = new Supplier { CompanyName = "Hello Kitty" };
        prodContext.Suppliers.Add(supplier);
    }
    supplier.Products.Add(product);
    prodContext.SaveChanges();

    var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
    Console.WriteLine("Supplier List:");
    {
        foreach(var s in data1){
            Console.WriteLine("Company name:");
            Console.WriteLine(s.CompanyName);
            Console.WriteLine("Available products:");
            foreach(var p in s.Products)
            {
                Console.WriteLine(p.Name);
            }
        }
    }
}
```

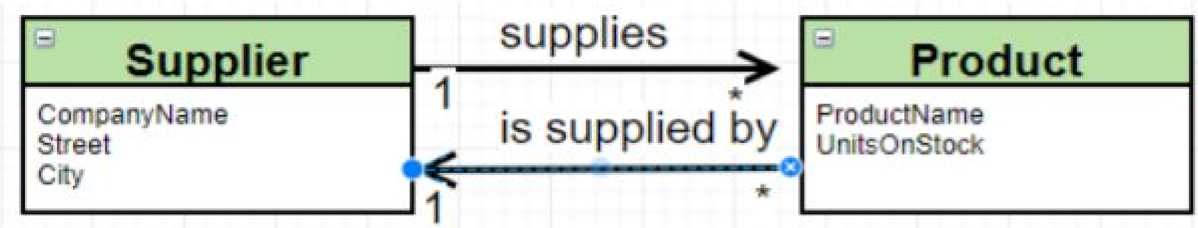
Wynik:



```
C:\Users\Andy\source\
Supplier List:
Company name:
Hello Kitty
Available products:
Kasza wielka
Zupa
Kasza wielka
```

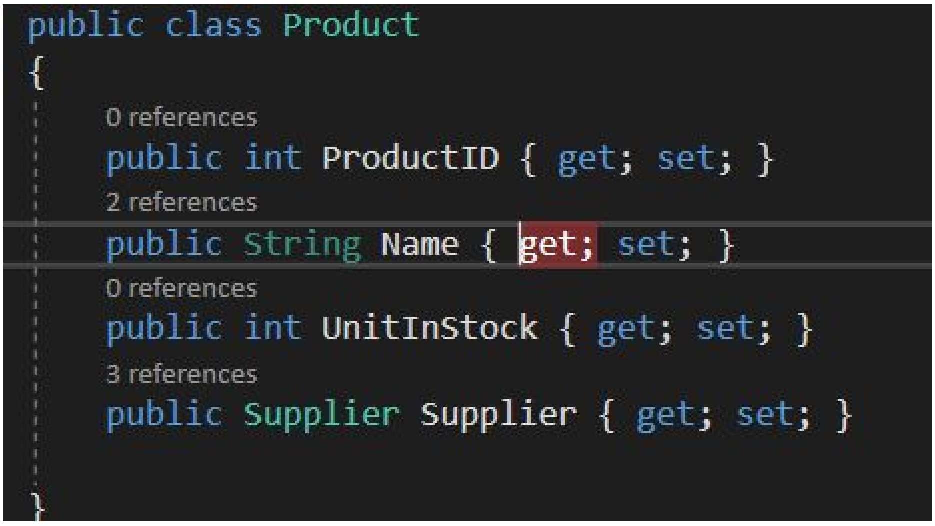
4.

IV. Zamodeluj relacje dwustronną jak poniżej:



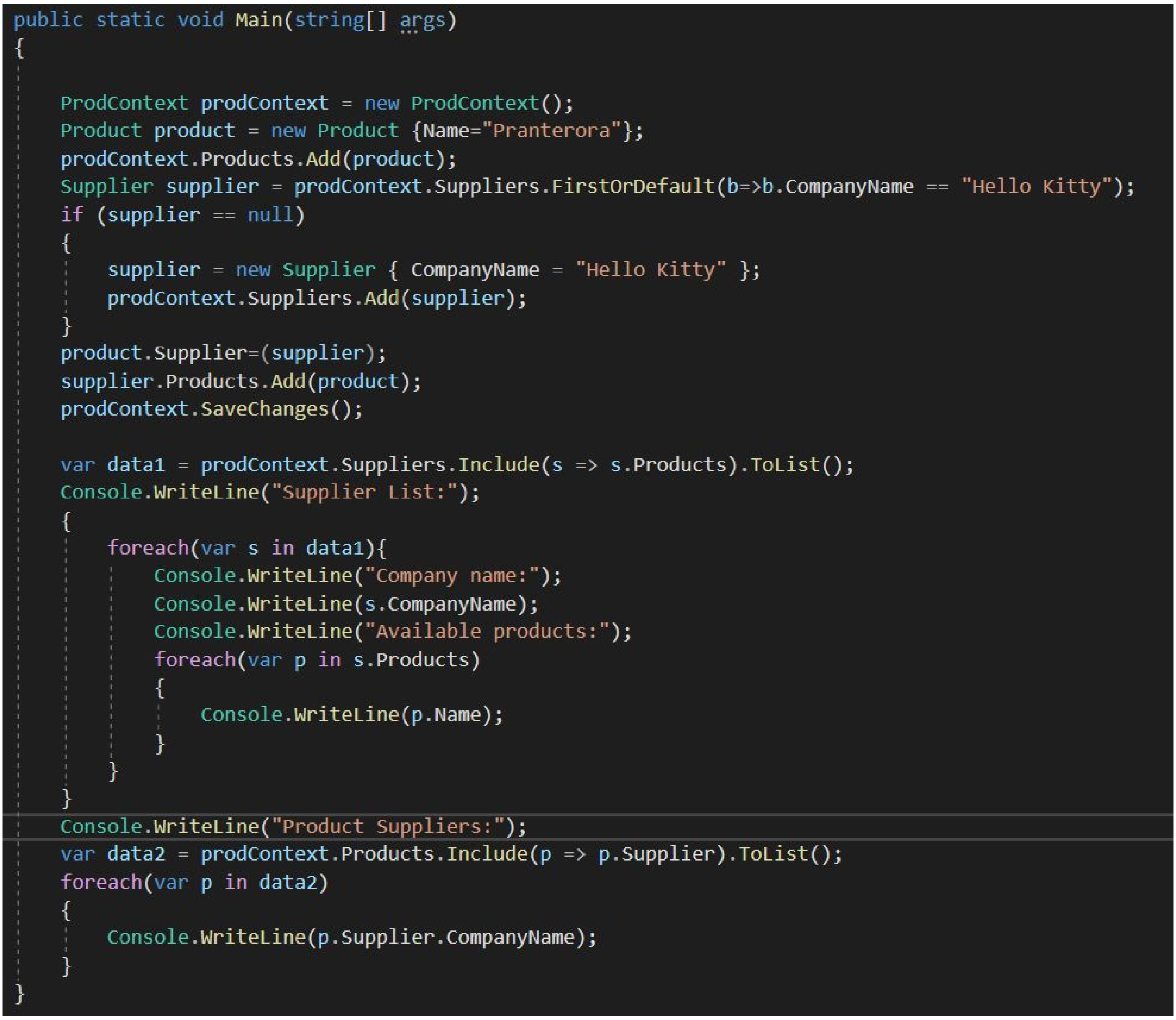
a. Tradycyjnie: Stworz kilka produktow

Dodałem atrybut Supplier do klasy Product



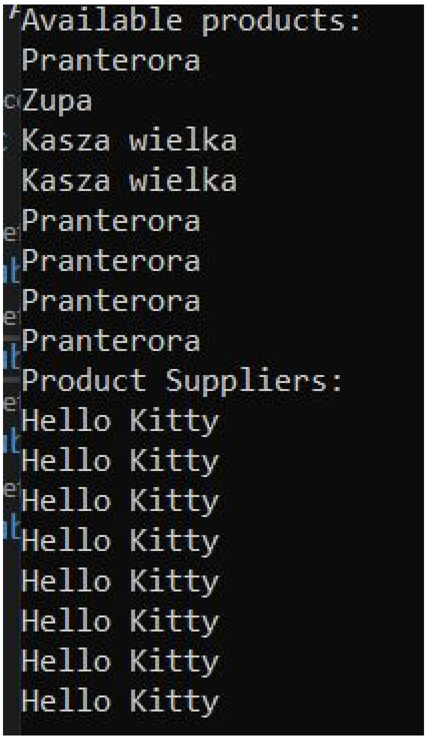
```
public class Product
{
    0 references
    public int ProductID { get; set; }
    2 references
    public String Name { get; set; }
    0 references
    public int UnitInStock { get; set; }
    3 references
    public Supplier Supplier { get; set; }

}
```

Modifikacja Program.cs żeby sprawdzić poprawność działania programu



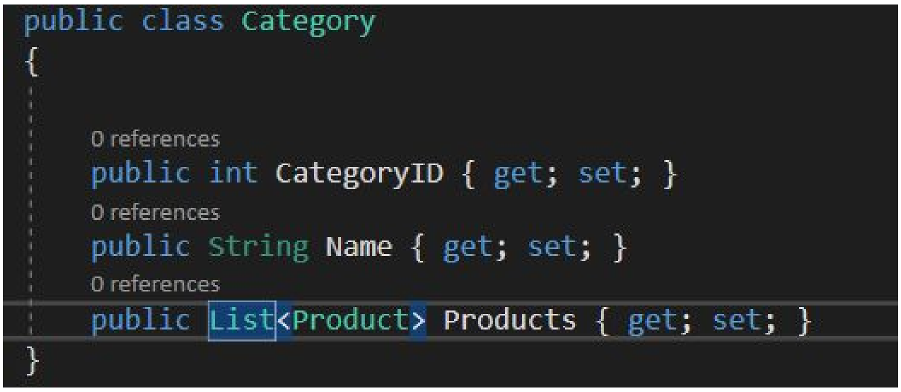
```
public static void Main(string[] args)
{

    ProdContext prodContext = new ProdContext();
    Product product = new Product {Name="Pranterora"};
    prodContext.Products.Add(product);
    Supplier supplier = prodContext.Suppliers.FirstOrDefault(b=>b.CompanyName == "Hello Kitty");
    if (supplier == null)
    {
        supplier = new Supplier { CompanyName = "Hello Kitty" };
        prodContext.Suppliers.Add(supplier);
    }
    product.Supplier=(supplier);
    supplier.Products.Add(product);
    prodContext.SaveChanges();

    var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
    Console.WriteLine("Supplier List:");
    {
        foreach(var s in data1){
            Console.WriteLine("Company name:");
            Console.WriteLine(s.CompanyName);
            Console.WriteLine("Available products:");
            foreach(var p in s.Products)
            {
                Console.WriteLine(p.Name);
            }
        }
    }
    Console.WriteLine("Product Suppliers:");
    var data2 = prodContext.Products.Include(p => p.Supplier).ToList();
    foreach(var p in data2)
    {
        Console.WriteLine(p.Supplier.CompanyName);
    }
}
```

Wynik:



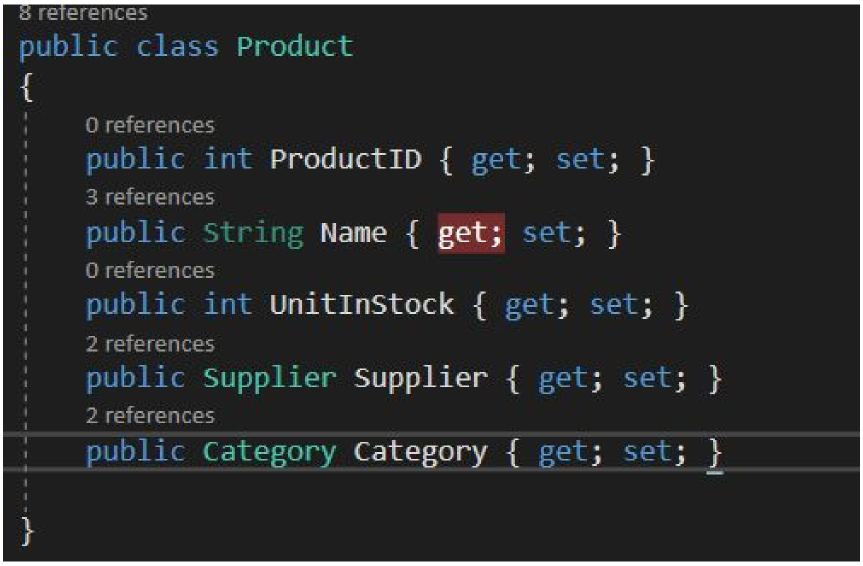
```
Available products:
Pranterora
Zupa
Kasza wielka
Kasza wielka
Pranterora
Pranterora
Pranterora
Pranterora
Product Suppliers:
Hello Kitty
Hello Kitty
Hello Kitty
Hello Kitty
Hello Kitty
Hello Kitty
Hello Kitty
Hello Kitty
```

5.Dodano klase Category z property int CategoryID, String Name oraz listą produktow



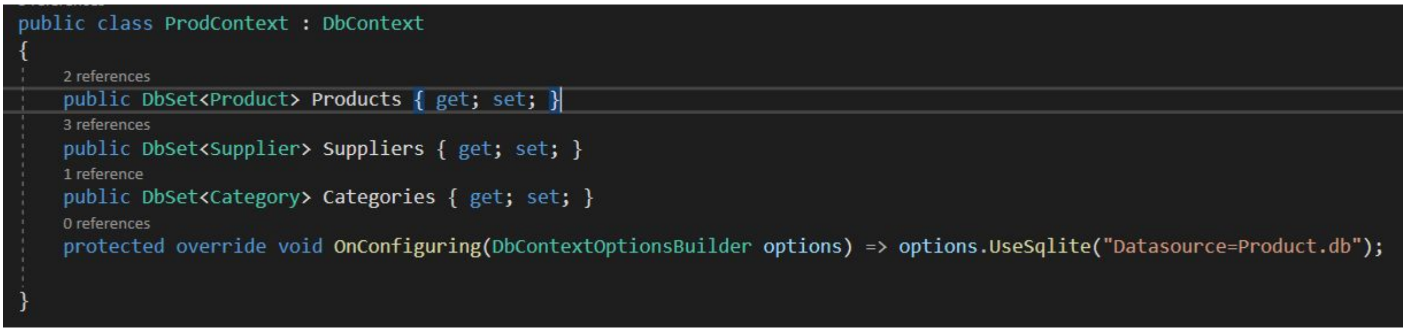
```
public class Category
{

    0 references
    public int CategoryID { get; set; }
    0 references
    public String Name { get; set; }
    0 references
    public List<Product> Products { get; set; }
}
```

Dodanie atrybutu Category do klasy Product



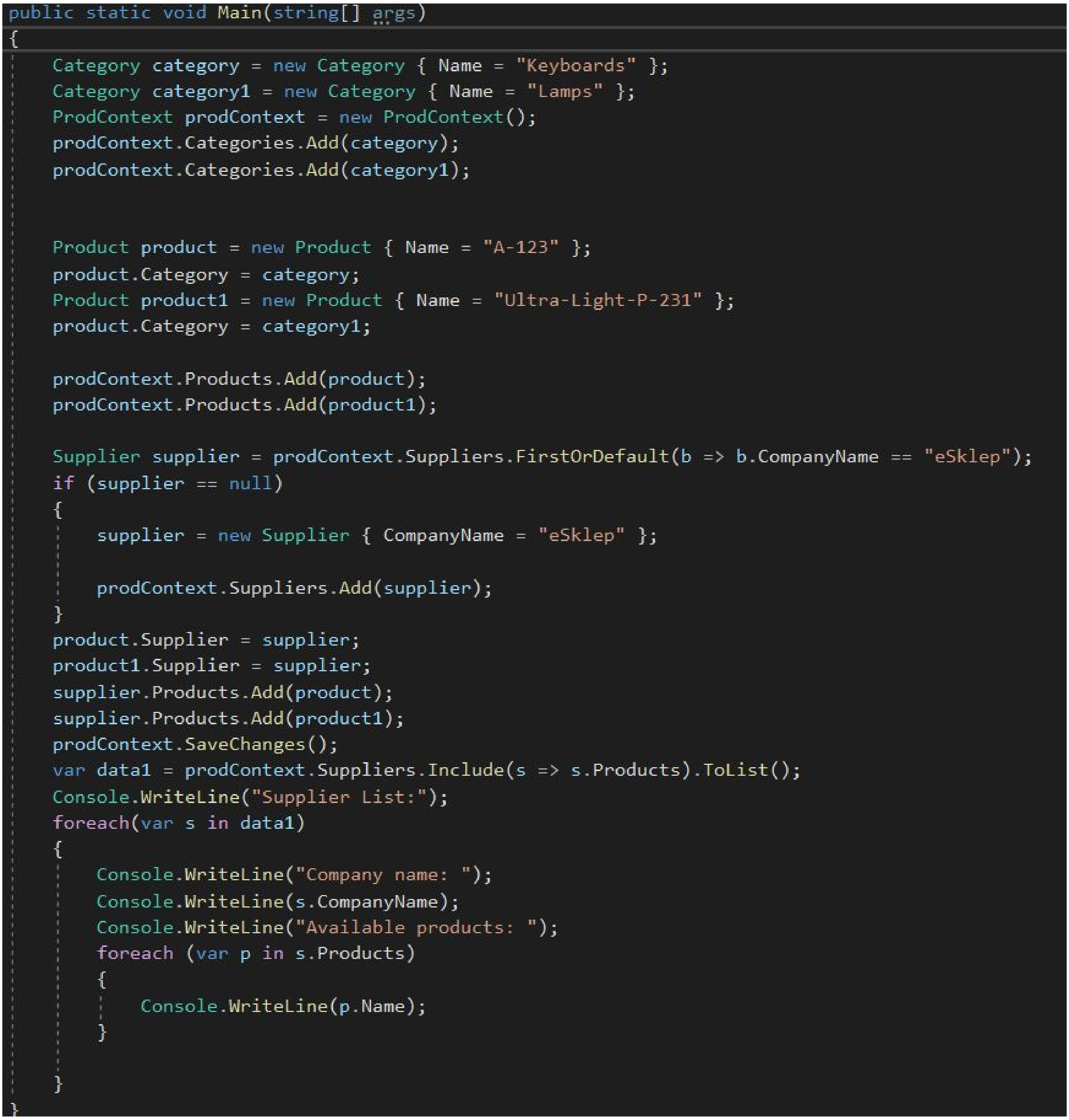
```
8 references
public class Product
{
    0 references
    public int ProductID { get; set; }
    3 references
    public String Name { get; set; }
    0 references
    public int UnitInStock { get; set; }
    2 references
    public Supplier Supplier { get; set; }
    2 references
    public Category Category { get; set; }


}
```

Dodanie listy Categories do prodContext



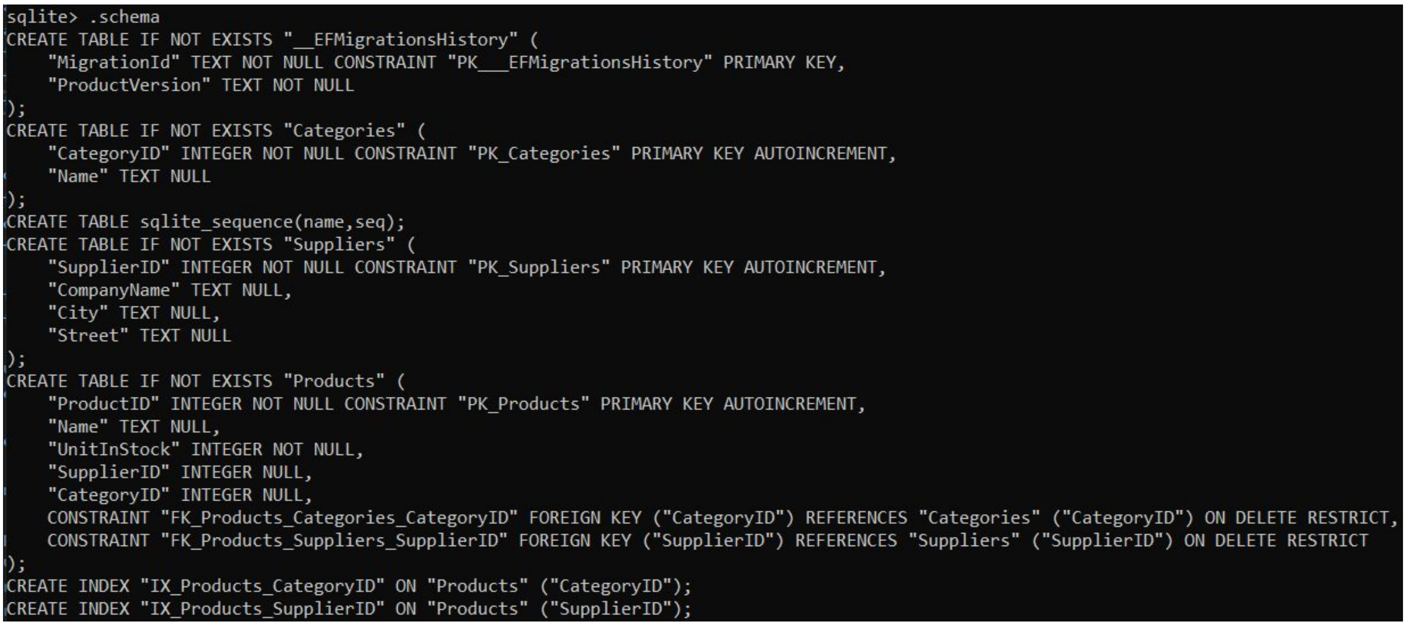
```
public class ProdContext : DbContext
{
    2 references
    public DbSet<Product> Products { get; set; }
    3 references
    public DbSet<Supplier> Suppliers { get; set; }
    1 reference
    public DbSet<Category> Categories { get; set; }
    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder options) => options.UseSqlite("Datasource=Product.db");

}
```

Modifikacja Program.cs żeby sprawdzić poprawność działania programu



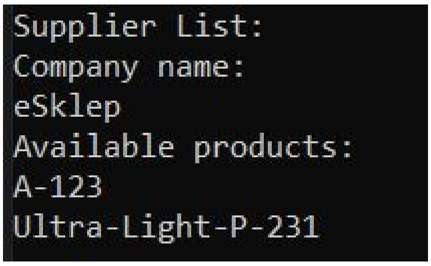
```
public static void Main(string[] args)
{
    Category category = new Category { Name = "Keyboards" };
    Category category1 = new Category { Name = "Lamps" };
    ProdContext prodContext = new ProdContext();
    prodContext.Categories.Add(category);
    prodContext.Categories.Add(category1);


    Product product = new Product { Name = "A-123" };
    product.Category = category;
    Product product1 = new Product { Name = "Ultra-Light-P-231" };
    product.Category = category1;

    prodContext.Products.Add(product);
    prodContext.Products.Add(product1);

    Supplier supplier = prodContext.Suppliers.FirstOrDefault(b => b.CompanyName == "eSklep");
    if (supplier == null)
    {
        supplier = new Supplier { CompanyName = "eSklep" };

        prodContext.Suppliers.Add(supplier);
    }
    product.Supplier = supplier;
    product1.Supplier = supplier;
    supplier.Products.Add(product);
    supplier.Products.Add(product1);
    prodContext.SaveChanges();
    var data1 = prodContext.Suppliers.Include(s => s.Products).ToList();
    Console.WriteLine("Supplier List:");
    foreach(var s in data1)
    {
        Console.WriteLine("Company name: ");
        Console.WriteLine(s.CompanyName);
        Console.WriteLine("Available products: ");
        foreach (var p in s.Products)
        {
            Console.WriteLine(p.Name);
        }

    }
}
```

Wynik widoczny w schemie:



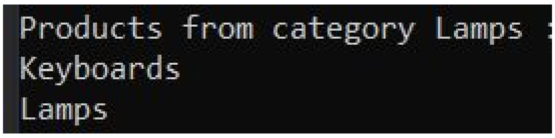
```
sqlite> .schema
CREATE TABLE IF NOT EXISTS "__EFMigrationsHistory" (
    "MigrationId" TEXT NOT NULL CONSTRAINT "PK___EFMigrationsHistory" PRIMARY KEY,
    "ProductVersion" TEXT NOT NULL
);
CREATE TABLE IF NOT EXISTS "Categories" (
    "CategoryID" INTEGER NOT NULL CONSTRAINT "PK_Categories" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL
);
CREATE TABLE sqlite_sequence(name,seq);
CREATE TABLE IF NOT EXISTS "Suppliers" (
    "SupplierID" INTEGER NOT NULL CONSTRAINT "PK_Suppliers" PRIMARY KEY AUTOINCREMENT,
    "CompanyName" TEXT NULL,
    "City" TEXT NULL,
    "Street" TEXT NULL
);
CREATE TABLE IF NOT EXISTS "Products" (
    "ProductID" INTEGER NOT NULL CONSTRAINT "PK_Products" PRIMARY KEY AUTOINCREMENT,
    "Name" TEXT NULL,
    "UnitInStock" INTEGER NOT NULL,
    "SupplierID" INTEGER NULL,
    "CategoryID" INTEGER NULL,
    CONSTRAINT "FK_Products_Categories_CategoryID" FOREIGN KEY ("CategoryID") REFERENCES "Categories" ("CategoryID") ON DELETE RESTRICT,
    CONSTRAINT "FK_Products_Suppliers_SupplierID" FOREIGN KEY ("SupplierID") REFERENCES "Suppliers" ("SupplierID") ON DELETE RESTRICT
);
CREATE INDEX "IX_Products_CategoryID" ON "Products" ("CategoryID");
CREATE INDEX "IX_Products_SupplierID" ON "Products" ("SupplierID");
```

Wynik:



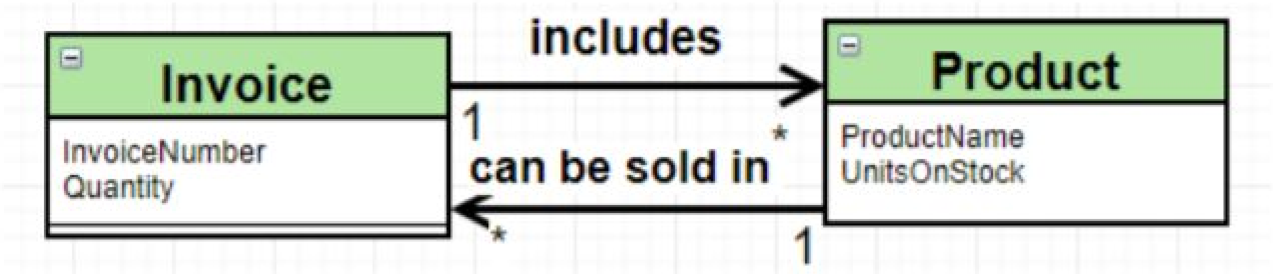
```
Supplier List:
Company name:
eSklep
Available products:
A-123
Ultra-Light-P-231
```

Otrzymanie produktu z kategorii:

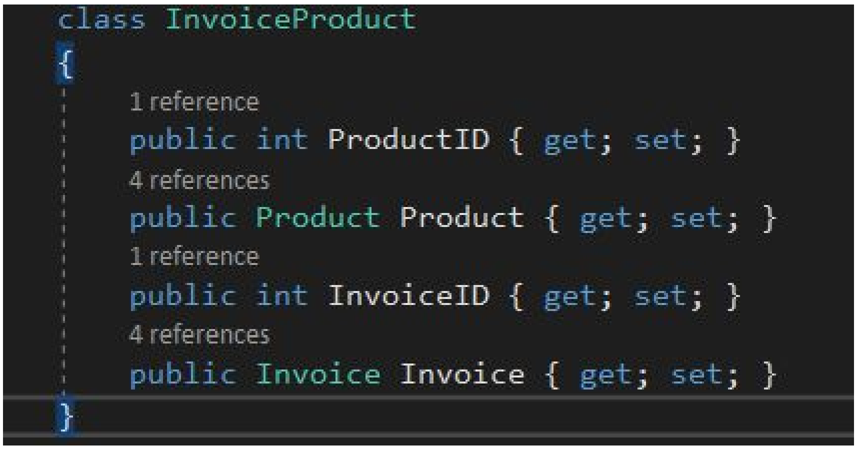


```
Products from category Lamps :
Keyboards
Lamps
```

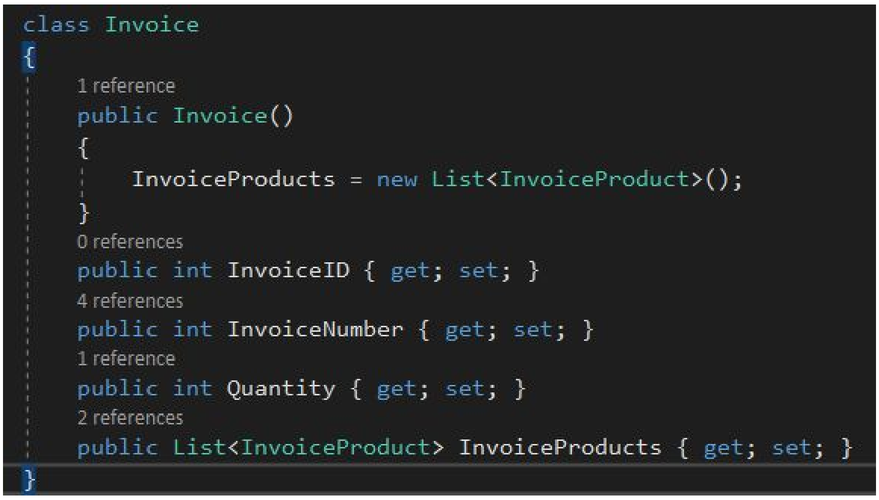
6.



Dodalem klase Invoice i InvoiceProduct

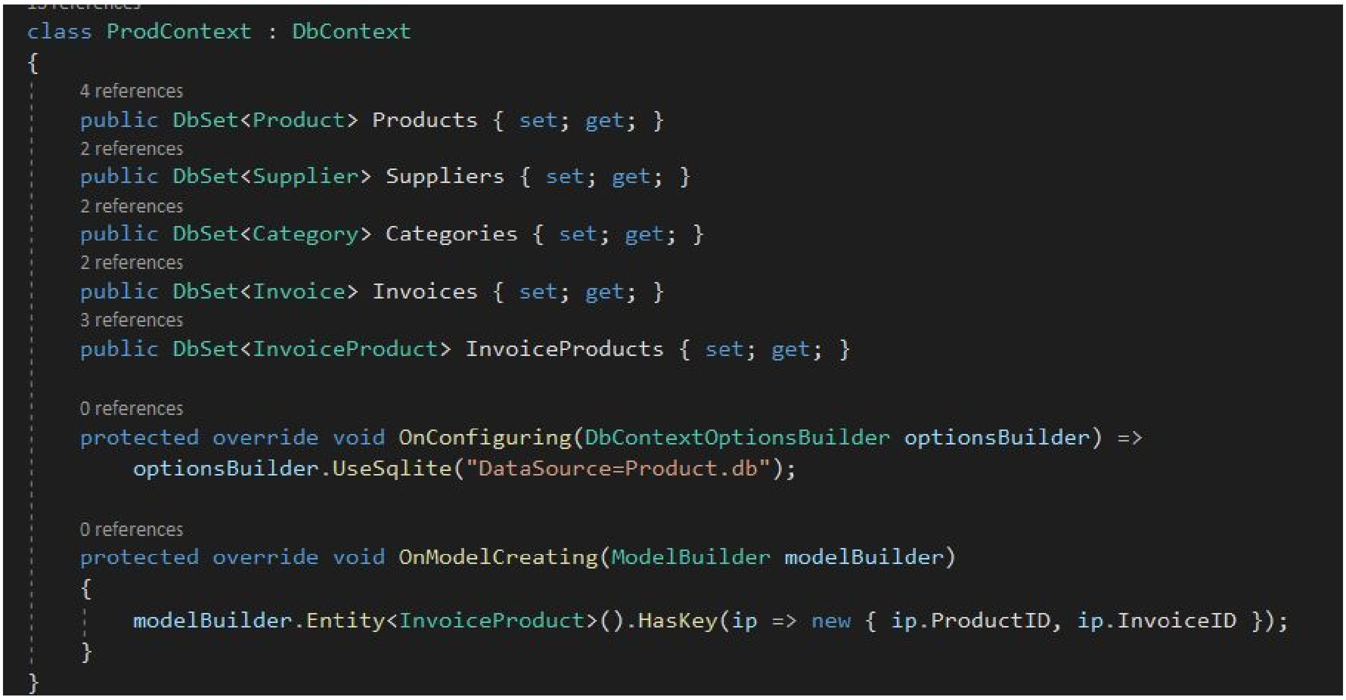


```
class InvoiceProduct
{
    1 reference
    public int ProductID { get; set; }
    4 references
    public Product Product { get; set; }
    1 reference
    public int InvoiceID { get; set; }
    4 references
    public Invoice Invoice { get; set; }
}
```

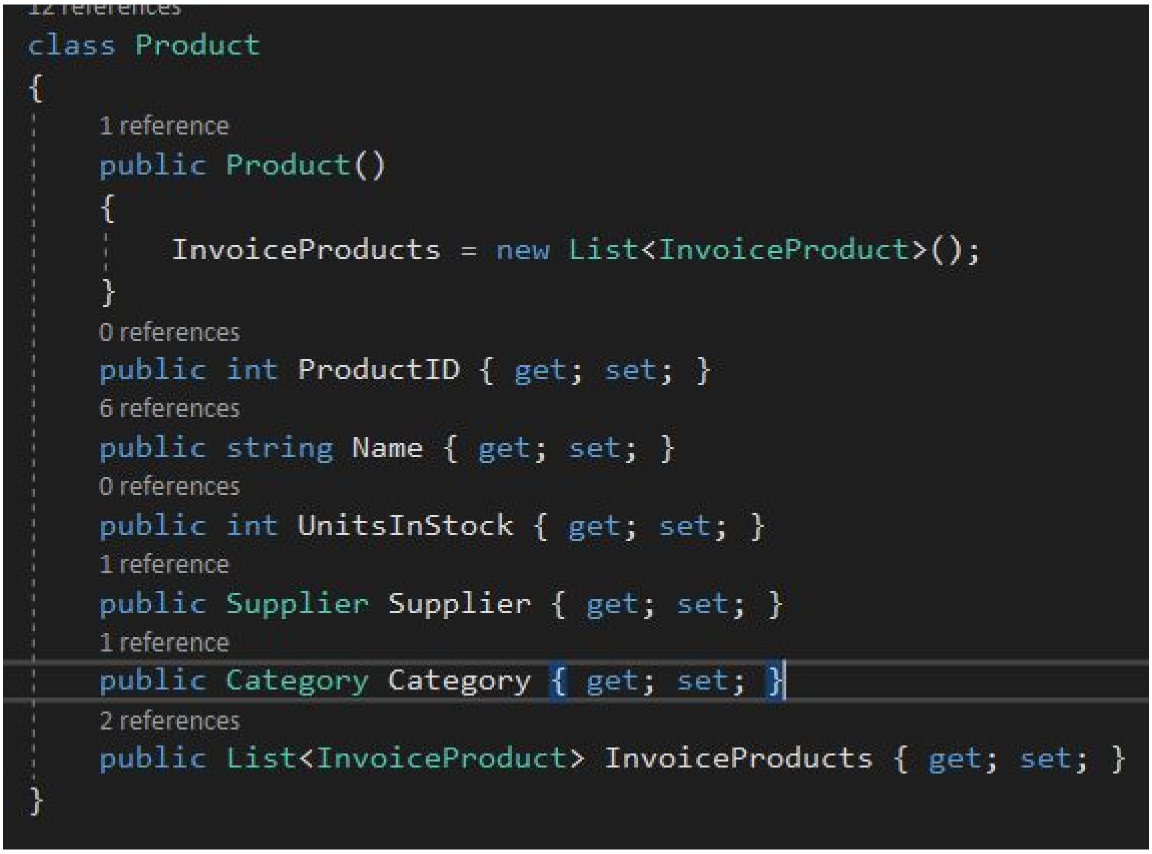


```
class Invoice
{
    1 reference
    public Invoice()
    {
        InvoiceProducts = new List<InvoiceProduct>();
    }
    0 references
    public int InvoiceID { get; set; }
    4 references
    public int InvoiceNumber { get; set; }
    1 reference
    public int Quantity { get; set; }
    2 references
    public List<InvoiceProduct> InvoiceProducts { get; set; }
}
```

Modyfikacja ProdContext i Products dla przechowywania Invoices

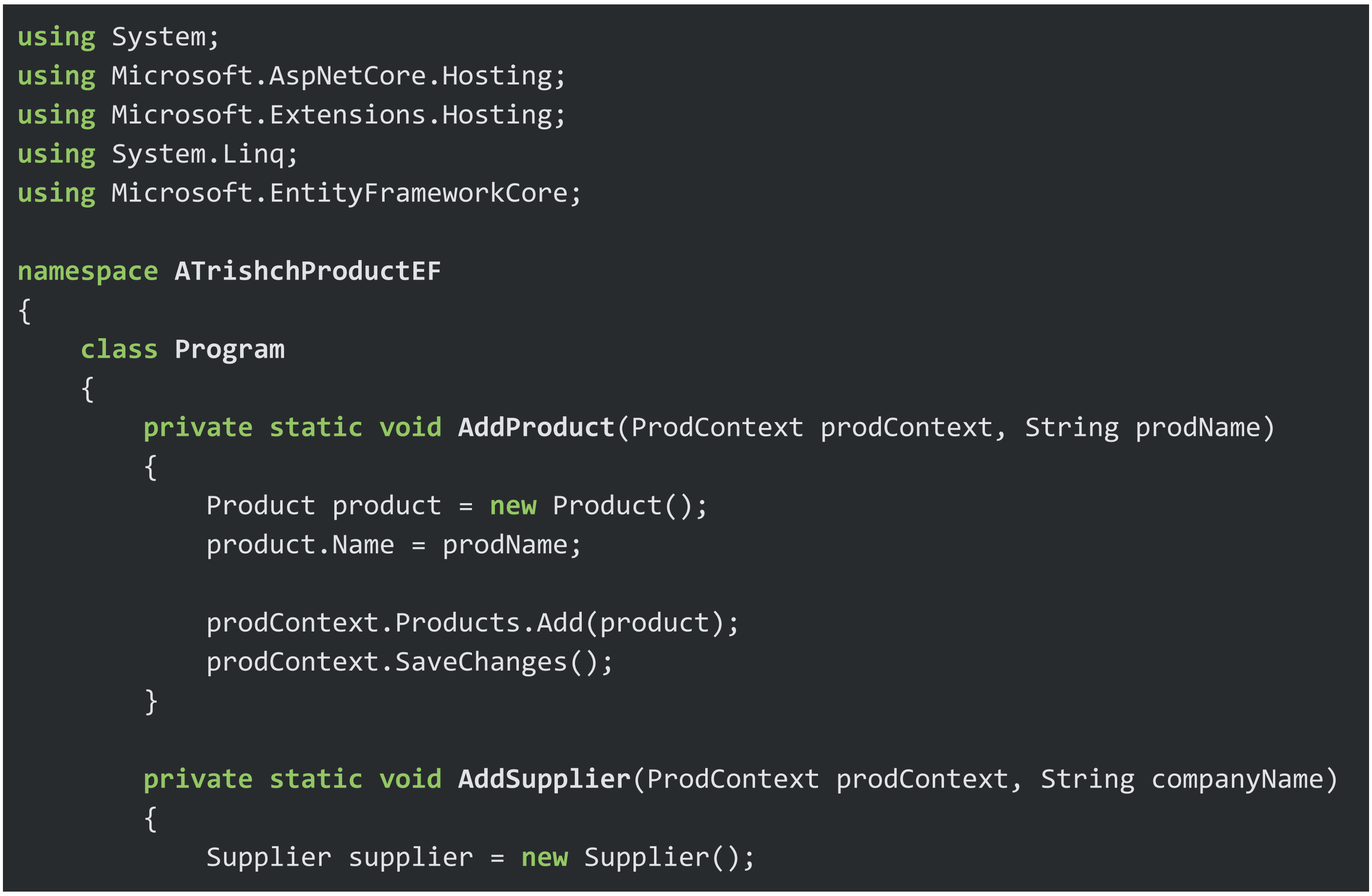


```
class ProdContext : DbContext
{
    4 references
    public DbSet<Product> Products { set; get; }
    2 references
    public DbSet<Supplier> Suppliers { set; get; }
    2 references
    public DbSet<Category> Categories { set; get; }
    2 references
    public DbSet<Invoice> Invoices { set; get; }
    3 references
    public DbSet<InvoiceProduct> InvoiceProducts { set; get; }

    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) =>
        optionsBuilder.UseSqlite("DataSource=Product.db");

    0 references
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<InvoiceProduct>().HasKey(ip => new { ip.ProductID, ip.InvoiceID });
    }
}
```



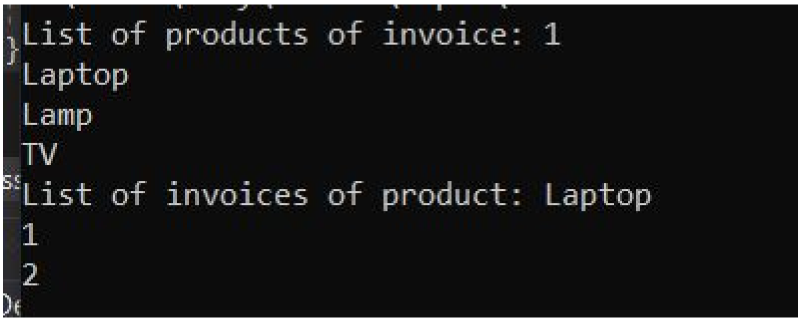
```
12 references
class Product
{
    1 reference
    public Product()
    {
        InvoiceProducts = new List<InvoiceProduct>();
    }
    0 references
    public int ProductID { get; set; }
    6 references
    public string Name { get; set; }
    0 references
    public int UnitsInStock { get; set; }
    1 reference
    public Supplier Supplier { get; set; }
    1 reference
    public Category Category { get; set; }
    2 references
    public List<InvoiceProduct> InvoiceProducts { get; set; }
}
```

Modifikacja Program.cs żeby sprawdzić poprawność działania programu,stworzone metody dla dodawania poszczególnych elementów i wyświetłiania danego invoicu



```
using System;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;
using System.Linq;
using Microsoft.EntityFrameworkCore;

namespace ATrishchProductEF
{
    class Program
    {
        private static void AddProduct(ProdContext prodContext, String prodName)
        {
            Product product = new Product();
            product.Name = prodName;

            prodContext.Products.Add(product);
            prodContext.SaveChanges();
        }

        private static void AddSupplier(ProdContext prodContext, String companyName)
        {
            Supplier supplier = new Supplier();
            supplier.CompanyName = companyName;

            prodContext.Suppliers.Add(supplier);
            prodContext.SaveChanges();
        }

        private static void AddCategory(ProdContext prodContext, String categoryName)
        {
            Category category = new Category();
            category.Name = categoryName;

            prodContext.Categories.Add(category);
            prodContext.SaveChanges();
        }

        private static void AddInvoice(ProdContext prodContext, int invoiceNumber,
int invoiceQuantity)
        {
            Invoice invoice = new Invoice();
            invoice.InvoiceNumber = invoiceNumber;
            invoice.Quantity = invoiceQuantity;

            prodContext.Invoices.Add(invoice);
            prodContext.SaveChanges();
        }

        private static void AddInvoiceProduct(ProdContext prodContext, int
invoiceNumber, String prodName)
        {
            Invoice invoice = prodContext.Invoices.Where(i => i.InvoiceNumber ==
invoiceNumber).FirstOrDefault();
            Product product = prodContext.Products.Where(p => p.Name ==
prodName).FirstOrDefault();

            InvoiceProduct invoiceProduct = new InvoiceProduct();
            invoiceProduct.Invoice = invoice;
            invoiceProduct.Product = product;

            invoice.InvoiceProducts.Add(invoiceProduct);
            product.InvoiceProducts.Add(invoiceProduct);

            prodContext.InvoiceProducts.Add(invoiceProduct);
            prodContext.SaveChanges();
        }
        private static void ConnectProductSupplier(ProdContext prodContext, String
prodName, String companyName)
        {
            Product product = prodContext.Products.Where(p => p.Name ==
prodName).FirstOrDefault();
            Supplier supplier = prodContext.Suppliers.Where(s => s.CompanyName ==
companyName).FirstOrDefault();

            supplier.Products.Add(product);
            product.Supplier = supplier;
            prodContext.SaveChanges();
        }

        private static void ConnectProductCategory(ProdContext prodContext, String
prodName, String categoryName)
        {
            Product product = prodContext.Products.Where(p => p.Name ==
prodName).FirstOrDefault();
            Category category = prodContext.Categories.Where(c => c.Name ==
categoryName).FirstOrDefault();

            category.Products.Add(product);
            product.Category = category;
            prodContext.SaveChanges();
        }

        private static void PrintProductsOfInvoice(ProdContext prodContext, int
invoiceNumber)
        {
            Console.WriteLine("List of products of invoice: " + invoiceNumber);

            var products = prodContext.InvoiceProducts
                .Include(ip => ip.Product)
                .Where(ip => ip.Invoice.InvoiceNumber == invoiceNumber)
                .Select(ip => ip.Product.Name).ToList();

            foreach (var p in products)
            {
                Console.WriteLine(p);
            }
        }

        private static void PrintInvoicesOfProduct(ProdContext prodContext, String
prodName)
        {
            Console.WriteLine("List of invoices of product: " + prodName);

            var invoices = prodContext.InvoiceProducts
                .Include(ip => ip.Invoice)
                .Where(ip => ip.Product.Name == prodName)
                .Select(ip => ip.Invoice.InvoiceNumber).ToList();

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

Wynik:



```
List of products of invoice: 1
Laptop
Lamp
TV
List of invoices of product: Laptop
1
2
```