Imiona i nazwiska autorów: Łukasz Kluza, Mateusz Sacha

Zadanie 1 - rozwiązanie

Product.cs

```
public class Product {
    public int ProductID { get; set; }
    public String? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

ProdContext.cs

```
public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("Datasource=MyProductDatabase");
    }
}
```

Program.cs

```
class Program {
    static void Main() {
        ProdContext prodContext = new ProdContext();
        Product product = CreateProduct();
        prodContext.Products.Add(product);
        prodContext.SaveChanges();

        var query = from prod in prodContext.Products
                    select prod.ProductName;

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

    private static Product CreateProduct(){
        Console.WriteLine("Write new product name: ");
        String? prodName = Console.ReadLine();
        Product product = new Product { ProductName = prodName };
        Console.WriteLine("Write new product units in stock: ");
```

```
            String? units = Console.ReadLine();
            if(units != null) {
                int prodUnits = Int32.Parse(units);
                product.UnitsInStock = prodUnits;
            }
            return product;
        }
    }
```

Przykład działania:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
Write new product name:
Myszka
Write new product units in stock:
24
Flamaster
Kreda
Pisak
Tablica
Kredki
Monitor
Klawiatura
Myszka
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab>
```

Diagram bazy danych:

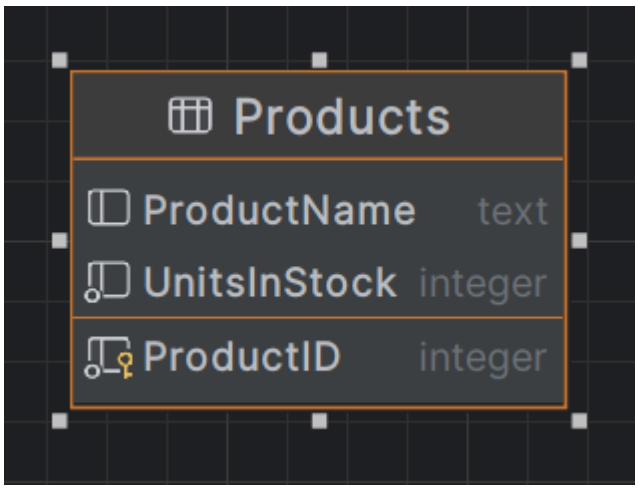


Tabela Products w bazie danych:

	ProductID	ProductName	UnitsInStock
1	1	Flamaster	150
2	2	Kreda	10
3	3	Pisak	15
4	4	Tablica	1
5	5	Kredki	190
6	6	Monitor	10
7	7	Klawiatura	15
8	8	Myszka	24

Zadanie 2 - rozwiązanie

a)

Product.cs

```
public class Product {
    public int ProductID { get; set; }
    public Supplier? Supplier { get; set; }
    public String? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

Suppliers.cs

```
public class Supplier {
    public int SupplierID { get; set; }
    public String? CompanyName { get; set; }
    public String? Street { get; set; }
    public String? City { get; set; }
}
```

ProdContext.cs

```
public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
```

```
        optionsBuilder.UseSqlite("Datasource=MyProductDatabase");
    }
}
```

Program.cs

```
class Program {
    static void Main() {
        ProdContext prodContext = new ProdContext();
        Console.WriteLine("Do you want to add new product (yes/no): ");
        String? response = Console.ReadLine();
        if(response == "yes") {
            Product product = CreateProduct();
            prodContext.Products.Add(product);
            prodContext.SaveChanges();
        }

        Supplier supplier = CreateSupplier();
        prodContext.Add(supplier);

        var lastProduct = prodContext.Products
            .OrderByDescending(prod => prod.ProductID)
            .FirstOrDefault();

        if(lastProduct != null) lastProduct.Supplier = supplier;

        prodContext.SaveChanges();

        var query = from prod in prodContext.Products
            select new
            {
                prod.ProductID,
                prod.ProductName,
                SupplierName = prod.Supplier != null ? prod.Supplier.CompanyName :
"Unknown"
            };

        foreach (var prod in query) {
            Console.WriteLine($"[{prod.ProductID}] | {prod.ProductName} |
{prod.SupplierName}");
        }
    }

    private static Product CreateProduct(){
        Console.WriteLine("Write new product name: ");
        String? prodName = Console.ReadLine();
        Product product = new Product { ProductName = prodName };
        Console.WriteLine("Write new product units in stock: ");
        String? units = Console.ReadLine();
        if(units != null) {
            int prodUnits = Int32.Parse(units);
```

```
            product.UnitsInStock = prodUnits;
        }
        return product;
    }

    private static Supplier CreateSupplier(){
        Console.WriteLine("Write new supplier name: ");
        String? suppName = Console.ReadLine();
        Supplier supplier = new Supplier { CompanyName = suppName };
        Console.WriteLine("Write new supplier street: ");
        String? suppStreet = Console.ReadLine();
        supplier.Street = suppStreet;
        Console.WriteLine("Write new supplier city: ");
        String? suppCity = Console.ReadLine();
        supplier.City = suppCity;
        return supplier;
    }
}
```

Przykład działania z dodawaniem nowego produktu:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
Do you want to add new product (yes/no):
yes
Write new product name:
Woda
Write new product units in stock:
200
Write new supplier name:
Muszynianka
Write new supplier street:
muszynianka1
Write new supplier city:
Muszyna
[1] | Flamaster | Amazon
[2] | Flamaster | Amazon
[3] | Flamaster | PandaBuy
[4] | Kredki | Amazon
[5] | Tablica | Amazon
[6] | Kreda | Amazon
[7] | Pisak | PandaBuy
[9] | Woda | Muszynianka
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab>
```

Przykład działania bez dodawania nowego produktu:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
Do you want to add new product (yes/no):
no
Write new supplier name:
Piwniczanka
Write new supplier street:
piwniczanka1
Write new supplier city:
Piwniczna
[1] | Flamaster | Amazon
[2] | Flamaster | Amazon
[3] | Flamaster | PandaBuy
[4] | Kredki | Amazon
[5] | Tablica | Amazon
[6] | Kreda | Amazon
[7] | Pisak | PandaBuy
[9] | Woda | Piwniczanka
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab>
```

Diagram bazy danych:

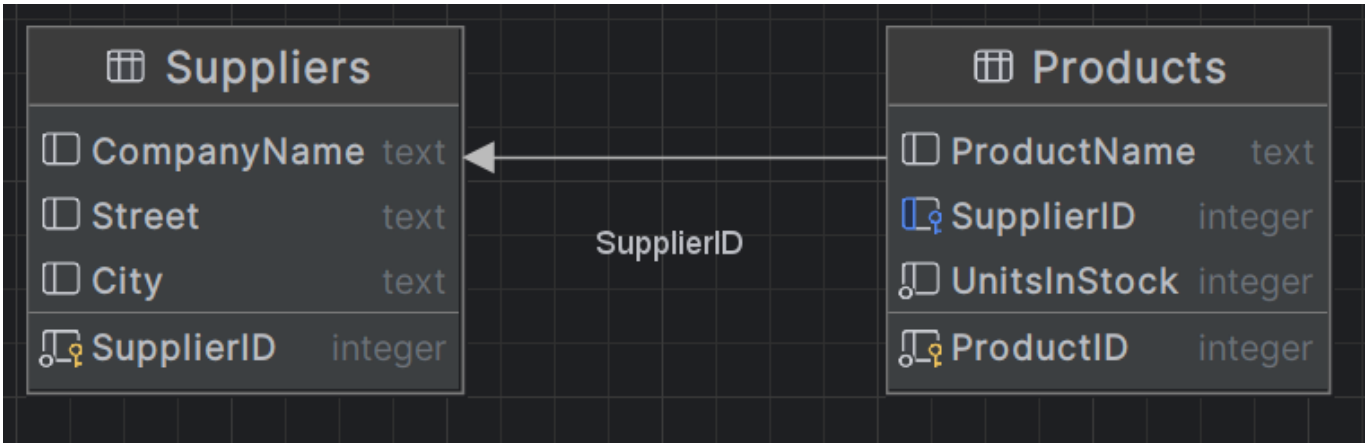


Tabela Products w bazie danych:

	ProductID	ProductName	SupplierID	UnitsInStock
1	1	Flamaster	1	7
2	2	Flamaster	2	10
3	3	Flamaster	3	1
4	4	Kredki	4	24
5	5	Tablica	4	25
6	6	Kreda	4	40
7	7	Pisak	5	0
8	9	Woda	9	200

Tabela Suppliers w bazie danych:

	SupplierID	CompanyName	Street	City
1	1	Amazon	amazon1	Berlin
2	2	Amazon	amazon2	Warsaw
3	3	PandaBuy	pandabuy1	NewYork
4	4	Amazon	amazon3	London
5	5	PandaBuy	pandabuy2	London
6	8	Muszynianka	muszynianka1	Muszyna
7	9	Piwniczanka	piwniczanka1	Piwniczna

---

b)

Product.cs

```
public class Product {
    public int ProductID { get; set; }
    public String? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

Supplier.cs

```
public class Supplier {
    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;} = new List<Product>();

    public override string ToString()
    {
        if (CompanyName != null)return CompanyName;
```

```
        else return "Unknow";
    }
}
```

Program.cs

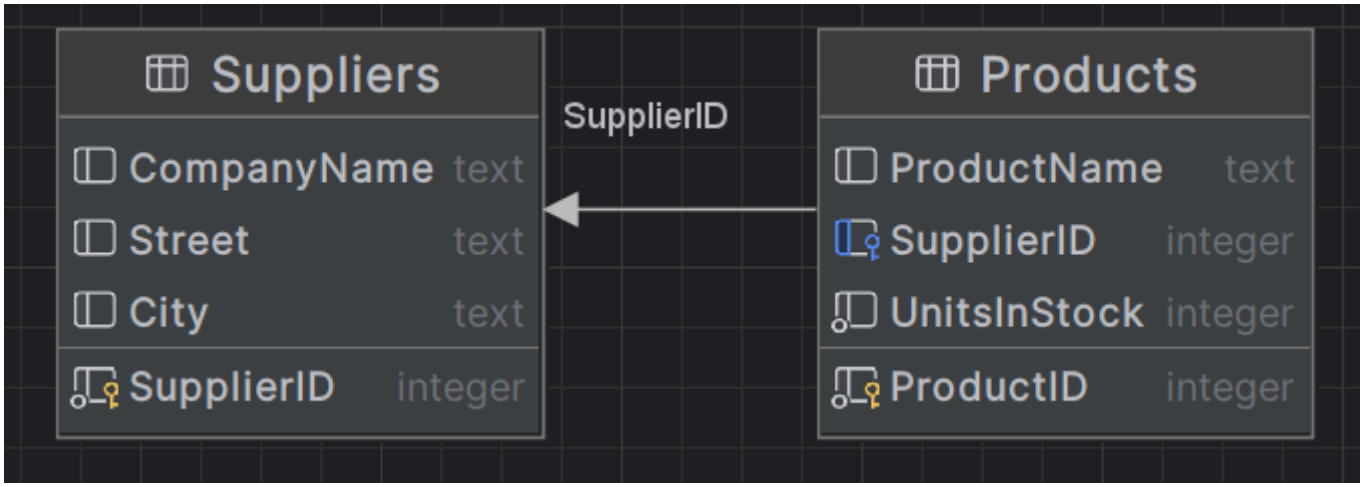
Funkcje CreateProduct oraz CreateSupplier nie uległy zmianie

```
class Program {
    static void Main() {
        ProdContext prodContext = new ProdContext();

        Supplier supplier = CreateSupplier();
        prodContext.Add(supplier);
        prodContext.SaveChanges();

        Console.WriteLine("How many products do you want to add: ");
        String? response = Console.ReadLine();
        int num = 0;
        if(response != null) num = Int32.Parse(response);

        while(num > 0) {
            Product product = CreateProduct();
            prodContext.Products.Add(product);
            supplier.Products.Add(product);
            prodContext.SaveChanges();
            num--;
        }

        var query = from prod in prodContext.Products
            select new
            {
                prod.ProductID,
                prod.ProductName,
            };

        foreach (var prod in query) {
            Console.WriteLine($"[{prod.ProductID}] | {prod.ProductName}");
        }
    }
}
```

Przykład działania:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
Write new supplier name:
Allegro
Write new supplier street:
allegro1
Write new supplier city:
Krakow
How many products do you want to add:
2
Write new product name:
Kreda
Write new product units in stock:
25
Write new product name:
Klawiatura
Write new product units in stock:
10
[1] | Flamaster
[2] | Flamaster
[3] | Flamaster
[4] | Kredki
[5] | Tablica
[6] | Kreda
[7] | Pisak
[9] | Woda
[10] | Kreda
[11] | Klawiatura
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab>
```

Diagram bazy danych:



Jak widać diagram nie uległ zmianie.

Tabela Products w bazie danych:

	ProductID	ProductName	SupplierID	UnitsInStock
1	1	Flamaster	1	7
2	2	Flamaster	2	10
3	3	Flamaster	3	1
4	4	Kredki	4	24
5	5	Tablica	4	25
6	6	Kreda	4	40
7	7	Pisak	5	0
8	9	Woda	9	200
9	10	Kreda	10	25
10	11	Klawiatura	10	10

Tabela Suppliers w bazie danych:

	SupplierID	CompanyName	Street	City
1	1	Amazon	amazon1	Berlin
2	2	Amazon	amazon2	Warsaw
3	3	PandaBuy	pandabuy1	NewYork
4	4	Amazon	amazon3	London
5	5	PandaBuy	pandabuy2	London
6	8	Muszynianka	muszynianka1	Muszyna
7	9	Piwniczanka	piwniczanka1	Piwniczna
8	10	Allegro	allegro1	Krakow

---

c)

Product.cs

```
public class Product {
    public int ProductID { get; set; }
    public Supplier? Supplier { get; set; }
    public String? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

Suppleir.cs

Klasa ta wygląda tak samo jak w punckie b.

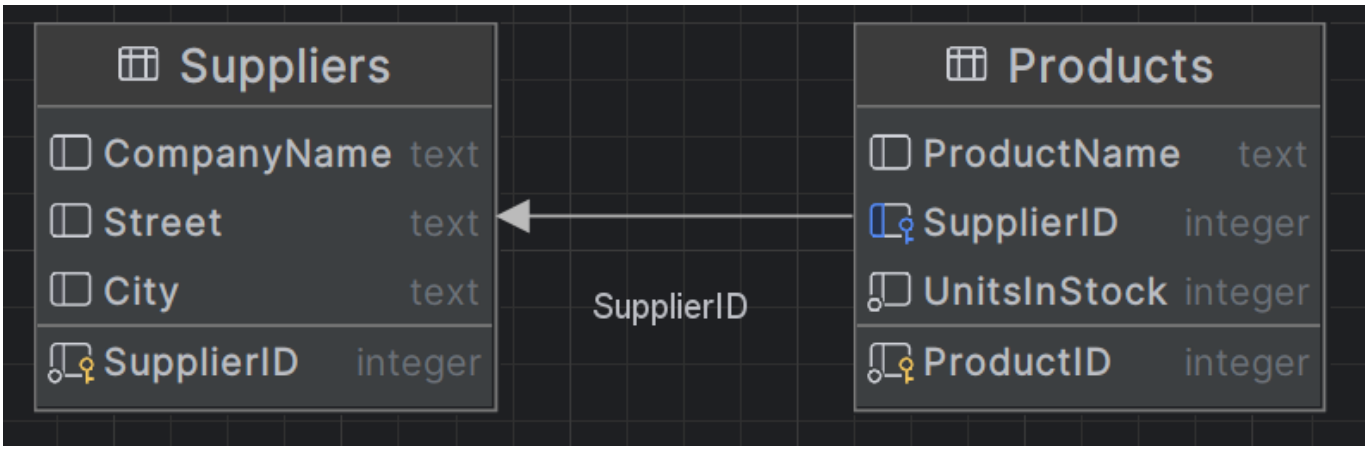
Program.cs

Klasa ta wygląda tak samo jak w punkcie b.

Przykład działania:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
Write new supplier name:
Olx
Write new supplier street:
olx1
Write new supplier city:
Katowice
How many products do you want to add:
2
Write new product name:
Monitor
Write new product units in stock:
15
Write new product name:
Mysz Komputerowa
Write new product units in stock:
33
[1] | Flamaster
[2] | Flamaster
[3] | Flamaster
[4] | Kredki
[5] | Tablica
[6] | Kreda
[7] | Pisak
[9] | Woda
[10] | Kreda
[11] | Klawiatura
[12] | Monitor
[13] | Mysz Komputerowa
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab>
```

Diagram bazy danych:



Jak widać diagram znów nie uległ zmianie.

Tabela Products w bazie danych:

	ProductID	ProductName	SupplierID	UnitsInStock
1	1	Flamaster	1	7
2	2	Flamaster	2	10
3	3	Flamaster	3	1
4	4	Kredki	4	24
5	5	Tablica	4	25
6	6	Kreda	4	40
7	7	Pisak	5	0
8	9	Woda	9	200
9	10	Kreda	10	25
10	11	Klawiatura	10	10
11	12	Monitor	11	15
12	13	Mysz Komputerowa	11	33

Tabela Suppliers w bazie danych:

	SupplierID	CompanyName	Street	City
1	1	Amazon	amazon1	Berlin
2	2	Amazon	amazon2	Warsaw
3	3	PandaBuy	pandabuy1	NewYork
4	4	Amazon	amazon3	London
5	5	PandaBuy	pandabuy2	London
6	8	Muszynianka	muszynianka1	Muszyna
7	9	Piwniczanka	piwniczanka1	Piwniczna
8	10	Allegro	allegro1	Krakow
9	11	Olx	olx1	Katowice

---

d)

Product.cs

```
public class Product {
    public int ProductID { get; set; }
    public virtual ICollection<InvoiceProduct>? InvoiceProducts { get; set; }
    public String? ProductName { get; set; }
    public int UnitsInStock { get; set; }
}
```

Invoice.cs

```
public class Invoice {
    [Key]
    public int InvoiceNumber { get; set; }
    public virtual ICollection<InvoiceProduct>? InvoiceProducts { get; set; }
```

InvoiceProduct.cs

```
public class InvoiceProduct {
    [Key, Column(Order = 0)]
    public int InvoiceID { get; set; }
    public virtual Invoice Invoice { get; set; }

    [Key, Column(Order = 1)]
    public int ProductID { get; set; }
    public virtual Product Product { get; set; }

    public int Quantity { get; set; }
}
}
```

ProdContext.cs

```
public class ProdContext : DbContext
{
    public DbSet<Product> Products { get; set; }
    public DbSet<Invoice> Invoices { get; set; }
    public DbSet<InvoiceProduct> InvoiceProducts { get; set; }

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

```
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("Datasource=MyProductDatabase");
    }
}
```

Program.cs

```
class Program
{
    static void Main()
    {
        ProdContext prodContext = new ProdContext();

        Console.WriteLine("How many products do you want to add: ");
        string? response = Console.ReadLine();
        int num = 0;
        if (response != null) num = Int32.Parse(response);

        while (num > 0)
        {
            Product product = CreateProduct();
            prodContext.Products.Add(product);
            prodContext.SaveChanges();
            num--;
        }

        Console.WriteLine("How many transactions do you want to add: ");
        response = Console.ReadLine();
        num = 0;
        if (response != null) num = Int32.Parse(response);

        while (num > 0)
        {
            Console.WriteLine("Listing all products with their units in stock");
            var query = from prod in prodContext.Products
                        select new
                        {
                            prod.ProductID,
                            prod.ProductName,
                            prod.UnitsInStock,
                        };

            foreach (var prod in query)
            {
                Console.WriteLine($"[{prod.ProductID}] | {prod.ProductName} |
{prod.UnitsInStock}");
            }
```

```
            Invoice invoice = CreateInvoice(prodContext);
            prodContext.Invoices.Add(invoice);
            prodContext.SaveChanges();
            num--;
        }

        Console.WriteLine("Enter the invoice number: ");
        int invoiceNumber = int.Parse(Console.ReadLine());
        ShowProductsSoldInInvoice(prodContext, invoiceNumber);

        Console.WriteLine("Enter the product ID: ");
        int productID = int.Parse(Console.ReadLine());
        ShowInvoicesForProduct(prodContext, productID);
    }

    private static Product CreateProduct()
    {
        Console.WriteLine("Write new product name: ");
        string? prodName = Console.ReadLine();
        Product product = new Product { ProductName = prodName };
        Console.WriteLine("Write new product units in stock: ");
        string? units = Console.ReadLine();
        if (units != null)
        {
            int prodUnits = Int32.Parse(units);
            product.UnitsInStock = prodUnits;
        }
        return product;
    }

    private static Invoice CreateInvoice(ProdContext prodContext)
    {
        Invoice invoice = new Invoice();
        invoice.InvoiceProducts = new List<InvoiceProduct>();

        Console.WriteLine("How many products do you want to buy: ");
        string? response = Console.ReadLine();
        int num = 0;
        if (response != null) num = Int32.Parse(response);

        while (num > 0)
        {
            Console.WriteLine("Write index of product you want to buy: ");
            string? index = Console.ReadLine();
            if (index != null)
            {
                int i = Int32.Parse(index);
                Console.WriteLine("Write how much of product you want to buy: ");
                string? count = Console.ReadLine();
                if (count != null)
                {
                    int c = Int32.Parse(count);
                    var product = prodContext.Products.FirstOrDefault(prod =>
prod.ProductID == i);
```

```
                    if (product != null)
                    {
                        if (product.UnitsInStock >= c)
                        {
                            InvoiceProduct invoiceProduct = new InvoiceProduct
                            {
                                Product = product,
                                Quantity = c
                            };
                            invoice.InvoiceProducts.Add(invoiceProduct);

                            product.UnitsInStock -= c;
                            Console.WriteLine("Product added to transaction.");
                            num--;
                        }
                        else
                        {
                            Console.WriteLine("There is not enough product units
in stock.");
                        }
                    }
                    else
                    {
                        Console.WriteLine("Product with this index doesn't
exist.");
                    }
                }
                else
                {
                    Console.WriteLine("That is not a number.");
                }
            }
        }

        return invoice;
    }

    static void ShowProductsSoldInInvoice(ProdContext prodContext, int
invoiceNumber)
    {
        var productsInInvoice = prodContext.InvoiceProducts
            .Include(ip => ip.Product)
            .Where(ip => ip.InvoiceNumber == invoiceNumber)
            .ToList();

        Console.WriteLine($"\nProducts sold within the invoice {invoiceNumber}:");
        foreach (var item in productsInInvoice)
        {
            Console.WriteLine($"- Product ID: {item.ProductID}, Name:
{item.Product.ProductName}, Quantity: {item.Quantity}");
        }
    }

    static void ShowInvoicesForProduct(ProdContext prodContext, int productID)
```

```
    {
        var invoicesForProduct = prodContext.InvoiceProducts
            .Include(ip => ip.Invoice)
            .Where(ip => ip.ProductID == productID)
            .Select(ip => ip.InvoiceNumber)
            .Distinct()
            .ToList();

        Console.WriteLine($"\nInvoices in which the product with ID {productID}
was sold:");
        foreach (var invoiceNumber in invoicesForProduct)
        {
            Console.WriteLine($"- Invoice Number: {invoiceNumber}");
        }
    }
}
```

Przykład działania:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
How many products do you want to add:
5
Write new product name:
Kredki
Write new product units in stock:
150
Write new product name:
Flamaster
Write new product units in stock:
100
Write new product name:
Kreda
Write new product units in stock:
10
Write new product name:
Tablica
Write new product units in stock:
5
Write new product name:
Papier
Write new product units in stock:
500
How many transactions do you want to add:
3
Listing all products with their units in stock
[1] | Kredki | 150
[2] | Flamaster | 100
```

```
[1] | Kredki | 150
[2] | Flamaster | 100
[3] | Kreda | 10
[4] | Tablica | 5
[5] | Papier | 500
How many products do you want to buy:
2
Write index of product you want to buy:
1
Write how much of product you want to buy:
100
Product added to transaction.
Write index of product you want to buy:
2
Write how much of product you want to buy:
50
Product added to transaction.
Listing all products with their units in stock
[1] | Kredki | 50
[2] | Flamaster | 50
[3] | Kreda | 10
[4] | Tablica | 5
[5] | Papier | 500
How many products do you want to buy:
3
Write index of product you want to buy:
4
Write how much of product you want to buy:
4
```

```
Write index of product you want to buy:
4
Write how much of product you want to buy:
4
Product added to transaction.
Write index of product you want to buy:
5
Write how much of product you want to buy:
490
Product added to transaction.
Write index of product you want to buy:
2
Write how much of product you want to buy:
25
Product added to transaction.
Listing all products with their units in stock
[1] | Kredki | 50
[2] | Flamaster | 25
[3] | Kreda | 10
[4] | Tablica | 1
[5] | Papier | 10
How many products do you want to buy:
1
Write index of product you want to buy:
1
Write how much of product you want to buy:
40
Product added to transaction.
```

```
Product added to transaction.
Listing all products with their units in stock
[1] | Kredki | 50
[2] | Flamaster | 25
[3] | Kreda | 10
[4] | Tablica | 1
[5] | Papier | 10
How many products do you want to buy:
1
Write index of product you want to buy:
1
Write how much of product you want to buy:
40
Product added to transaction.
Enter the invoice number:
1

Products sold within the invoice 1:
- Product ID: 1, Name: Kredki, Quantity: 100
- Product ID: 2, Name: Flamaster, Quantity: 50
Enter the product ID:
1

Invoices in which the product with ID 1 was sold:
- Invoice Number: 1
- Invoice Number: 3
```

Diagram bazy danych:

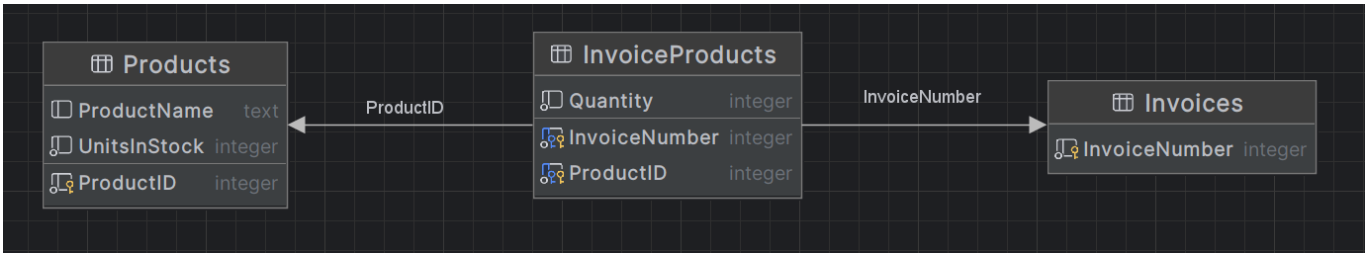


Tabela Products w bazie danych:

	ProductID	ProductName	UnitsInStock
1	1	Kredki	10
2	2	Flamaster	25
3	3	Kreda	10
4	4	Tablica	1
5	5	Papier	10

Tabela Invoices w bazie danych:

	InvoiceNumber
1	1
2	2
3	3

Tabela InvoiceProducts w bazie danych:

	InvoiceNumber	ProductID	Quantity
1	1	1	100
2	1	2	50
3	2	2	25
4	2	4	4
5	2	5	490
6	3	1	40

---

e)

Company.cs

```
public abstract class Company {
    public int CompanyID { get; set; }
    public String? CompanyName { get; set; }
    public String? Street { get; set; }
    public String? City { get; set; }
    public String? ZipCode { get; set; }

    public override string ToString()
    {
        if (CompanyName != null)return CompanyName;
```

```
        else return "Unknow";
    }
}
```

Supplier.cs

```
public class Supplier : Company {
    public String? BankAccountNumber { get; set; }
}
```

Customer.cs

```
public class Customer : Company {
    public float Discount { get; set; }
}
```

ProdContext.cs

```
public class ProdContext : DbContext
{
    public DbSet<Company>? Companies { get; set; }
    public DbSet<Supplier>? Suppliers { get; set; }
    public DbSet<Customer>? Customers { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        optionsBuilder.UseSqlite("Datasource=MyProductDatabase");
    }
}
```

Program.cs

```
class Program
{
    static void Main()
    {
        ProdContext prodContext = new ProdContext();

        Console.WriteLine("How many products do you want to add: ");
        string? response = Console.ReadLine();
        int num = 0;
        if (response != null) num = Int32.Parse(response);
```

```
        while(num>0)
        {
            Console.WriteLine("Enter company type (1 for Supplier, 2 for
Customer): ");
            string typeInput = Console.ReadLine();
            if (typeInput == "1")
            {
                var supplier = new Supplier();
                GetCompanyDetails(supplier);
                Console.WriteLine("Enter bank account number: ");
                supplier.BankAccountNumber = Console.ReadLine();
                prodContext.Companies.Add(supplier);
            }
            else if (typeInput == "2")
            {
                var customer = new Customer();
                GetCompanyDetails(customer);
                Console.WriteLine("Enter discount: ");
                if (float.TryParse(Console.ReadLine(), out float discount))
                {
                    customer.Discount = discount;
                }
                else
                {
                    Console.WriteLine("Invalid discount, setting to 0.");
                    customer.Discount = 0;
                }
                prodContext.Companies.Add(customer);
            }
            else
            {
                Console.WriteLine("Invalid company type.");
                return;
            }

            prodContext.SaveChanges();
            num--;
        }

        Console.WriteLine("Companies added successfully.");

        var companies = prodContext.Companies.ToList();
        foreach (var company in companies)
        {
            Console.WriteLine($"Company ID: {company.CompanyID}, Name:
{company.CompanyName}, Type: {company.GetType().Name}");
        }
    }

    static void GetCompanyDetails(Company company)
    {
        Console.WriteLine("Enter company name:");
        company.CompanyName = Console.ReadLine();
```

```
            Console.WriteLine("Enter street:");
            company.Street = Console.ReadLine();
            Console.WriteLine("Enter city:");
            company.City = Console.ReadLine();
            Console.WriteLine("Enter zip code:");
            company.ZipCode = Console.ReadLine();
        }
    }
```

Przykład działania:

```
PS C:\Users\zsuet\OneDrive\Pulpit\MSachaEFLab> dotnet run
How many products do you want to add:
5
Enter company type (1 for Supplier, 2 for Customer):
1
Enter company name:
Amazon
Enter street:
AmazonStreet
Enter city:
London
Enter zip code:
32500
Enter bank account number:
59999433999
Enter company type (1 for Supplier, 2 for Customer):
1
Enter company name:
PandaBuy
Enter street:
PandaStreet
Enter city:
Tokio
Enter zip code:
85599
Enter bank account number:
99932323233
Enter company type (1 for Supplier, 2 for Customer):
```

```
Enter bank account number:
99932323233
Enter company type (1 for Supplier, 2 for Customer):
2
Enter company name:
Nokia
Enter street:
NokiaStreet
Enter city:
Paris
Enter zip code:
58993
Enter discount:
0.25
Invalid discount, setting to 0.
Enter company type (1 for Supplier, 2 for Customer):
2
Enter company name:
Samsung
Enter street:
SamsungStreet
Enter city:
Bratyslava
Enter zip code:
88833
Enter discount:
0,5
Enter company type (1 for Supplier, 2 for Customer):
1
```

```
Enter discount:
0,5
Enter company type (1 for Supplier, 2 for Customer):
1
Enter company name:
Allegro
Enter street:
AlleStreet
Enter city:
Warsaw
Enter zip code:
32690
Enter bank account number:
99232323212
Companies added successfully.
Company ID: 1, Name: Amazon, Type: Supplier
Company ID: 2, Name: PandaBuy, Type: Supplier
Company ID: 3, Name: Nokia, Type: Customer
Company ID: 4, Name: Samsung, Type: Customer
Company ID: 5, Name: Allegro, Type: Supplier
```

Diagram bazy danych:

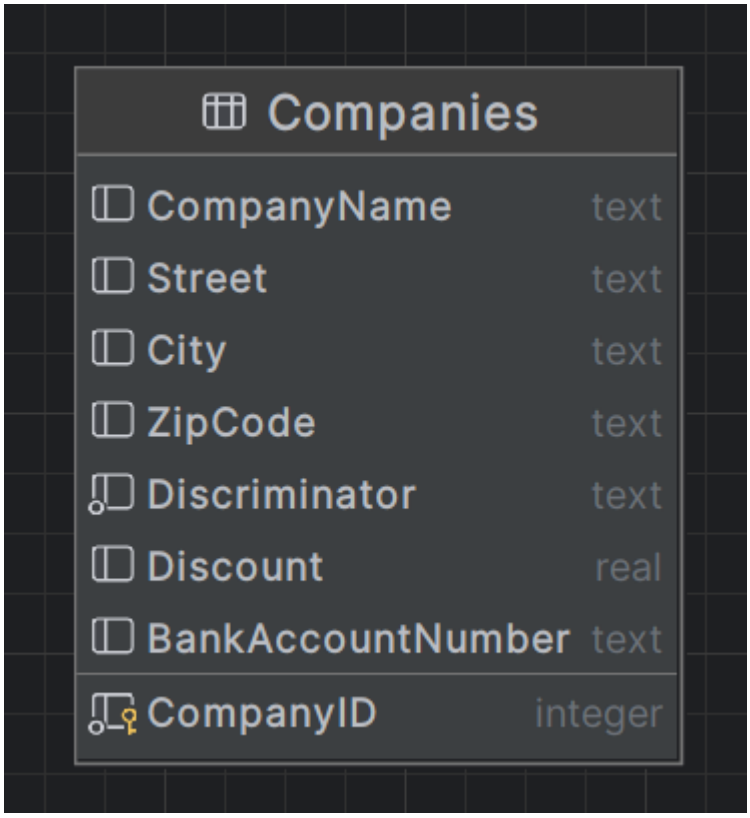


Tabela Companies w bazie danych:

	CompanyID	CompanyName	Street	City	ZipCode	Discriminator	Discount	BankAccountNumber
1	1	Amazon	AmazonStreet	London	32500	Supplier	<null>	59999433999
2	2	PandaBuy	PandaStreet	Tokio	85599	Supplier	<null>	99932323233
3	3	Nokia	NokiaStreet	Paris	58993	Customer	0	<null>
4	4	Samsung	SamsungStreet	Bratyslava	88833	Customer	0.5	<null>
5	5	Allegro	AlleStreet	Warsaw	32690	Supplier	<null>	99232323212

---

f)

Company.cs

```
public class Company {
    public int CompanyID { get; set; }
    public String? CompanyName { get; set; }
    public String? Street { get; set; }
    public String? City { get; set; }
    public String? ZipCode { get; set; }

    public override string ToString()
    {
        if (CompanyName != null)return CompanyName;
        else return "Unknow";
    }
}
```

Supplier.cs

Taki sam jak w punkcie e.

Customer.cs

Taki sam jak w punkcie e.

ProdContext.cs

```
public class ProdContext : DbContext
{
    public DbSet<Company> Companies { get; set; }
    public DbSet<Supplier> Suppliers { get; set; }
    public DbSet<Customer> Customers { get; set; }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Company>().UseTptMappingStrategy();
    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
```

```
            optionsBuilder.UseSqlite("Datasource=MyProductDatabase");
        }
    }
```

Program.cs

Taki sam jak w punkcie e.

Przykład działania:

```
How many companies do you want to add:
5
Enter company type (1 for Supplier, 2 for Customer):
1
Enter company name:
Amazon
Enter street:
AmStreet
Enter city:
London
Enter zip code:
45553
Enter bank account number:
23323232323
Enter company type (1 for Supplier, 2 for Customer):
1
Enter company name:
XKom
Enter street:
xCOMStreet
Enter city:
Lublin
Enter zip code:
545555
Enter bank account number:
32312313213
Enter company type (1 for Supplier, 2 for Customer):
```

```
32312313213
Enter company type (1 for Supplier, 2 for Customer):
2
Enter company name:
McDonalds
Enter street:
McStreet
Enter city:
McCity
Enter zip code:
56666
Enter discount:
0,45
Enter company type (1 for Supplier, 2 for Customer):
2
Enter company name:
Coral
Enter street:
CorolStreet
Enter city:
Zakopane
Enter zip code:
54533
Enter discount:
0.15
Invalid discount, setting to 0.
Enter company type (1 for Supplier, 2 for Customer):
2
```

```
Invalid discount, setting to 0.
Enter company type (1 for Supplier, 2 for Customer):
2
Enter company name:
Apple
Enter street:
AppStreet
Enter city:
Madrit
Enter zip code:
53221
Enter discount:
0,15
Companies added successfully.
Company ID: 1, Name: Amazon, Type: Supplier
Company ID: 2, Name: XKom, Type: Supplier
Company ID: 3, Name: McDonalds, Type: Customer
Company ID: 4, Name: Coral, Type: Customer
Company ID: 5, Name: Apple, Type: Customer
```

Diagram bazy danych:

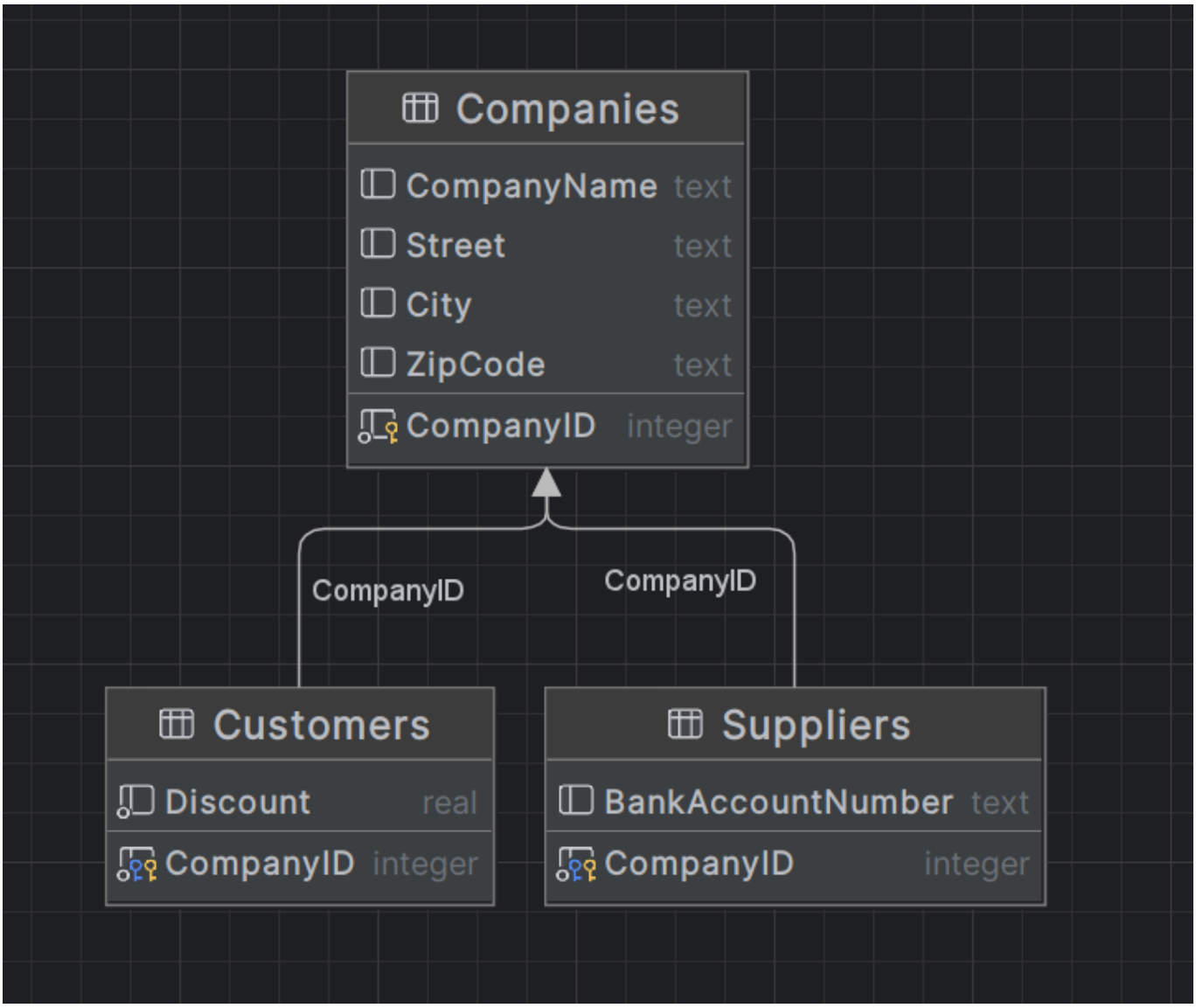


Tabela Companies w bazie danych:

	CompanyID	CompanyName	Street	City	ZipCode
1	1	Amazon	AmStreet	London	45553
2	2	XKom	xCOMStreet	Lublin	545555
3	3	McDonalds	McStreet	McCity	56666
4	4	Coral	CorolStreet	Zakopane	54533
5	5	Apple	AppStreet	Madrit	53221

Tabela Suppliers w bazie danych:

	CompanyID	BankAccountNumber
1	1	23323232323
2	2	32312313213

Tabela Customers w bazie danych:

	CompanyID	Discount
1	3	0.44999998807907104
2	4	0
3	5	0.15000000596046448

---

g)

Table-Per-Type (TPT):

- **Opis:** W strategii TPT, każda klasa w hierarchii dziedziczenia odpowiada osobnej tabeli w bazie danych. Dodatkowo, klasa podstawowa również ma swoją własną tabelę. Każda tabela podtypów zawiera kolumny odpowiadające właściwościom tego podtypu, a także klucz obcy do tabeli typów podstawowych, który identyfikuje odpowiedni wiersz w tabeli typów podstawowych dla każdego wiersza podtypu.

Table-Per-Hierarchy (TPH):

- **Opis:** W strategii TPH wszystkie klasy w hierarchii dziedziczenia mapowane są do jednej tabeli w bazie danych. Tabela ta zawiera kolumny dla wszystkich właściwości wszystkich klas w hierarchii, a dodatkowy wskaźnik typu określa typ rekordu.

Porównanie:

- **TPH:**
    - Mniejsza liczba tabel - dzięki temu, że wszystkie dane są przechowywane w jednej tabeli, struktura bazy danych jest prostsza.
    - Mniejsza klarowność danych - w przypadku dużej ilości klas w hierarchii, tabela może stać się bardziej złożona i trudniejsza do analizy.
    - Redundancja danych - niektóre kolumny dla określonych klas jednostek zawierają wartości NULL, a liczba tych kolumn zależy od liczby klas w hierarchii.
- **TPT:**
    - Rozdzielenie danych - każda tabela przechowuje dane tylko dla jednej konkretnej klasy, co prowadzi do klarownego przechowywania danych.
    - Łatwe dodawanie nowych klas - można łatwo dodać nowe klasy do hierarchii dziedziczenia, a EF Core automatycznie utworzy dla nich odpowiednie tabele.
    - Duża liczba tabel - struktura bazy jest bardziej złożona i powoduje to zmniejszenie wydajności operacji CRUD
- **Podsumowanie:** Oba podejścia dziedziczenia mają swoje wady i zalety, preferencja wyboru powinna raczej zależeć od tego jaką strukturę bazy danych uznamy za bardziej efektywną w danym projekcie.