

Uniwersytet WSB Merito

Programowanie Obiektowe

Ćwiczenia 4 sprawozdanie

Imiona i Nazwiska: Markin Vadym, Kachalov Yevhenii

Adresy mail: kachalovz171@gmail.com, markin4097@gmail.com

Nr.Albumów: 143069, 143071

Link na git: <https://github.com/BlupiR/Shop.git>

Tytuł projektu: Prosty system zarządzania sklepem

Opis: Twoim zadaniem jest stworzenie prostej konsolowej aplikacji do zarządzania sklepem.

Aplikacja powinna pozwalać na dodawanie produktów do sklepu, wyświetlanie listy dostępnych produktów oraz sprzedaż produktów do klientów.

Szczegółowe wymagania: 1. Utwórz klasę **Product** z właściwościami: **Name**, **Price**, **Quantity**.



```
1 using SampleHierarchies.Interfaces;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SampleHierarchies.Data
9 {
10     [reference]
11     public class Product : IProduct
12     {
13         [reference]
14         public string Name { get; set; }
15         [reference]
16         public double Price { get; set; }
17         [reference]
18         public int Quantity { get; set; }
19
20         public Product(string name, double price, int quantity)
21         {
22             Name = name;
23             Price = price;
24             Quantity = quantity;
25         }
26     }
27 }
```

2. Utwórz klasę **Shop** z listą produktów oraz metodami pozwalającymi na dodawanie produktów do listy, wyświetlanie dostępnych produktów oraz sprzedaż produktów.

```
1 using SampleHierarchies.Interfaces;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7 using Newtonsoft.Json;
8 using System.ComponentModel.Design;
9
10 namespace SampleHierarchies.Data
11 {
12     [reference]
13     public class Shop : IShop
14     {
15         [reference]
16         public List<IProduct> Product { get; set; }
17
18         [reference]
19         public Shop()
20         {
21             Product = new List<IProduct>();
22         }
23
24         //This method allows you to add products
25         [reference]
26         public void AddProduct(IProduct product)
27         {
28             if (product == null)
29             {
30                 throw new ArgumentNullException(nameof(product));
31             }
32
33             Product.Add(product);
34             Console.WriteLine($"Added product: {product.Name}");
35         }
36
37         //This method allows you to add products with discount
38         [reference]
39         public void AddDiscountedProduct(IProduct product)
40         {
41             //Implementation
42         }
43     }
44 }
```

3. Utwórz klasę **Customer** z metodą **Buy**, która umożliwi klientowi zakup produktu ze sklepu.

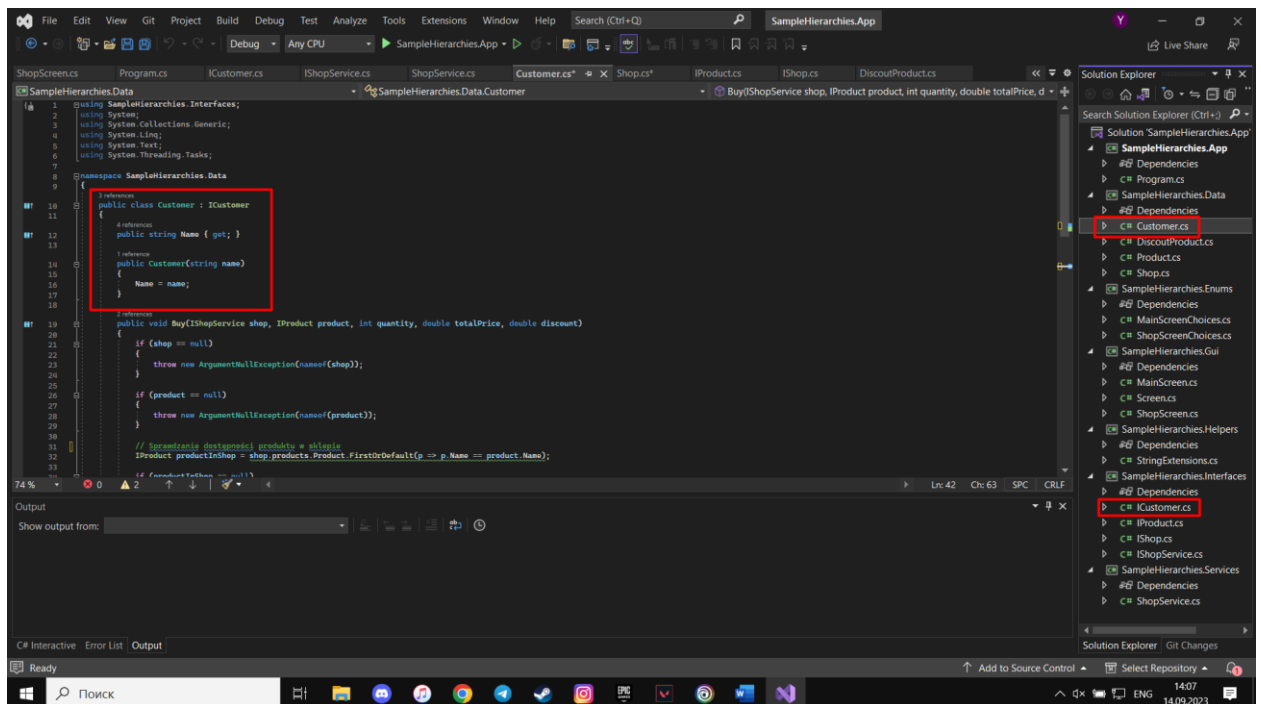
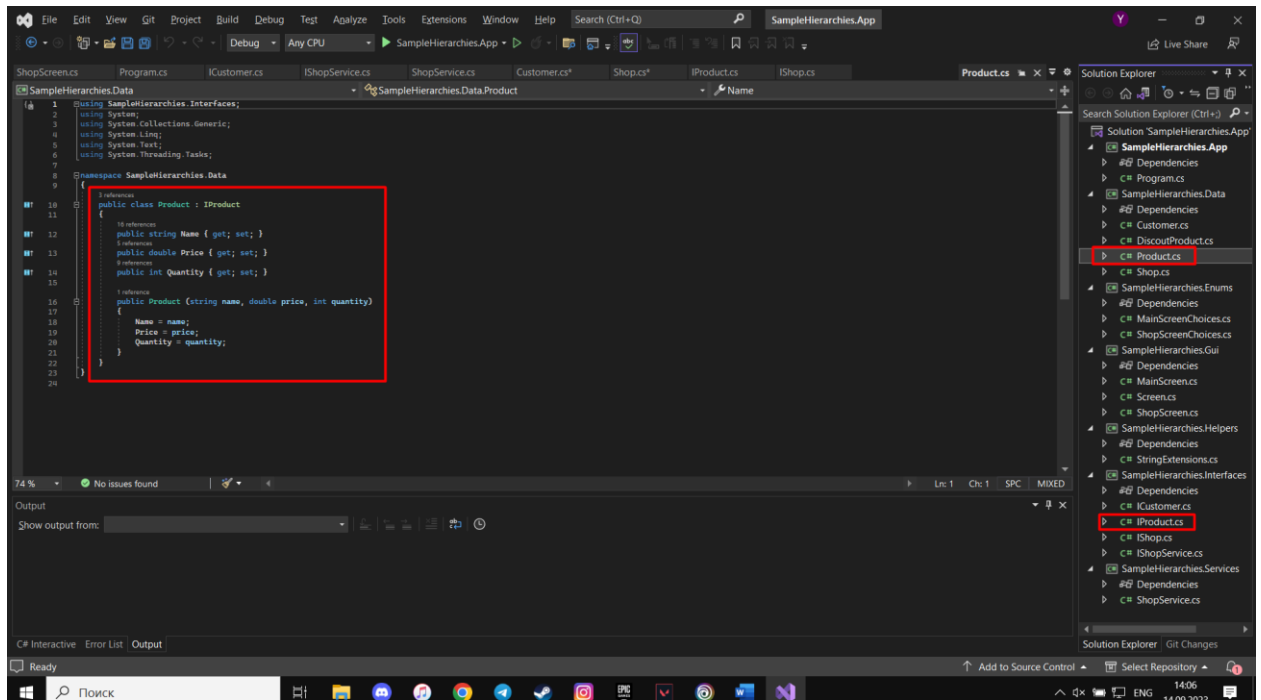


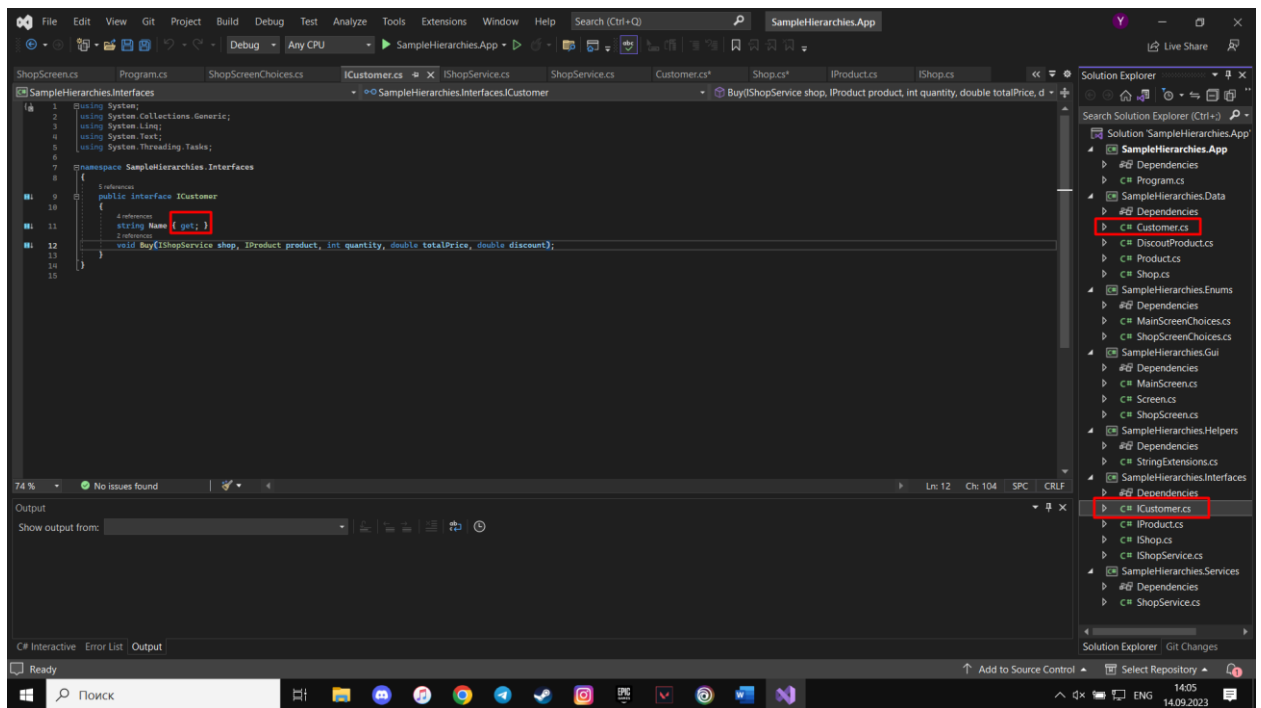
```
1 using SampleHierarchies.Interfaces;
2 using System;
3 using System.Collections.Generic;
4 using System.Linq;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SampleHierarchies.Data
9 {
10     public class Customer : ICustomer
11     {
12         public string Name { get; }
13
14         public Customer(string name)
15         {
16             Name = name;
17         }
18
19         public void Buy(IShopService shop, IProduct product, int quantity, double totalPrice, double discount)
20         {
21             if (shop == null)
22             {
23                 throw new ArgumentException(nameof(shop));
24             }
25
26             if (product == null)
27             {
28                 throw new ArgumentException(nameof(product));
29             }
30
31             // Sprawdzenie dostępności produktu w sklepie
32             IProduct productInShop = shop.products.Product.FirstOrDefault(p => p.Name == product.Name);
33             if (productInShop == null)
34             {
35                 throw new ArgumentException("Produkt nie jest dostępny w sklepie");
36             }
37         }
38     }
39 }
```

4. Zastosuj zasadę enkapsulacji - pola klasy powinny być prywatne, a dostęp do nich powinien być zapewniony poprzez publiczne metody **get** i **set**.

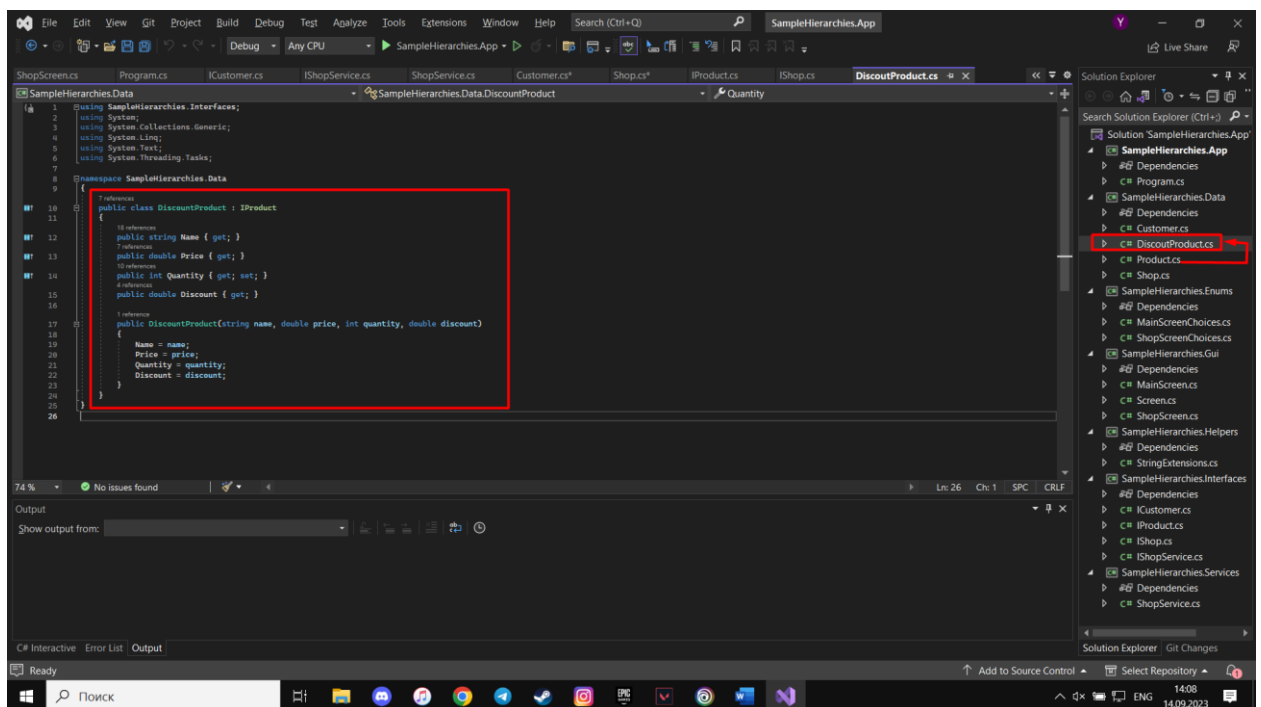


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Security.Cryptography;
5 using System.Text;
6 using System.Threading.Tasks;
7
8 namespace SampleHierarchies.Interfaces
9 {
10     public interface IShop
11     {
12         List<IProduct> Product { get; set; }
13         void AddProduct(IProduct product);
14         void ShowProducts();
15         void SellProduct(IProduct product, IShopService shopService, ICustomer customer);
16     }
17 }
18
```

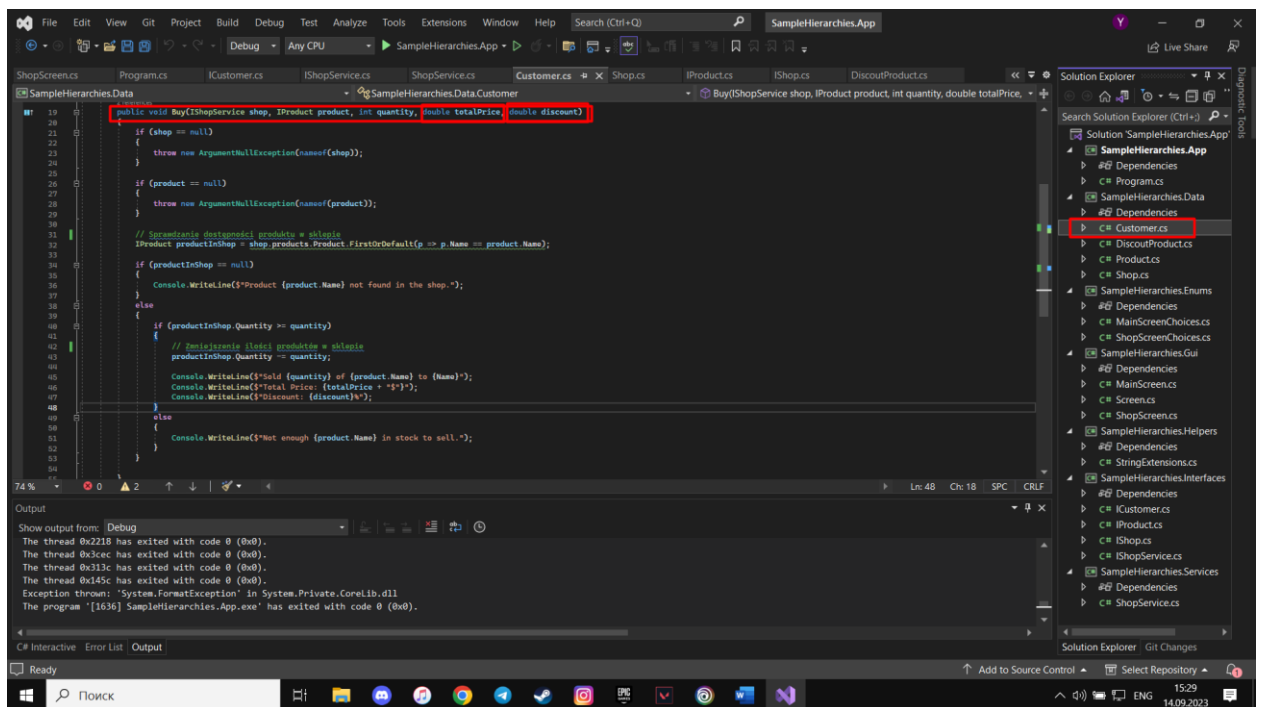




5. Zastosuj zasadę dziedziczenia - utwórz klasę **Discount Product**, która dziedziczy po klasie **IProduct** i dodaje nową właściwość **Discount**. ✓



6. Zastosuj zasadę polimorfizmu - metoda **Buy** w klasie **Customer** powinna inaczej obsługiwać zwykłe produkty i produkty ze zniżką. ✓



```
19 public void Buy(IShopService shop, IProduct product, int quantity, double totalPrice, double discount)
20 {
21     if (shop == null)
22     {
23         throw new ArgumentNullException(nameof(shop));
24     }
25
26     if (product == null)
27     {
28         throw new ArgumentNullException(nameof(product));
29     }
30
31     // Sprawdzenie dostawcy produktu = sklep
32     IProduct productInShop = shop.products.Product.FirstOrDefault(p => p.Name == product.Name);
33
34     if (productInShop == null)
35     {
36         Console.WriteLine($"Product {product.Name} not found in the shop.");
37     }
38     else
39     {
40         if (productInShop.Quantity >= quantity)
41         {
42             // Zmniejszenie ilości produktów = sklep
43             productInShop.Quantity -= quantity;
44
45             Console.WriteLine($"Sold {quantity} of {product.Name} to {Name}");
46             Console.WriteLine($"Total Price: {totalPrice + "5"}");
47             Console.WriteLine($"Discount: {discount}");
48         }
49         else
50         {
51             Console.WriteLine($"Not enough {product.Name} in stock to sell.");
52         }
53     }
54 }
```

Output

```
Show output from: Debug
The thread 0x2218 has exited with code 0 (0x0).
The thread 0x3cec has exited with code 0 (0x0).
The thread 0x313c has exited with code 0 (0x0).
The thread 0x145c has exited with code 0 (0x0).
Exception thrown: 'System.FormatException' in System.Private.CoreLib.dll
The program '[1636] SampleHierarchies.App.exe' has exited with code 0 (0x0).
```