

Programowanie Obiektowe

Oleksander Mikhov,

Yehor Vysotskyi

Radeo30001@gmail.com,

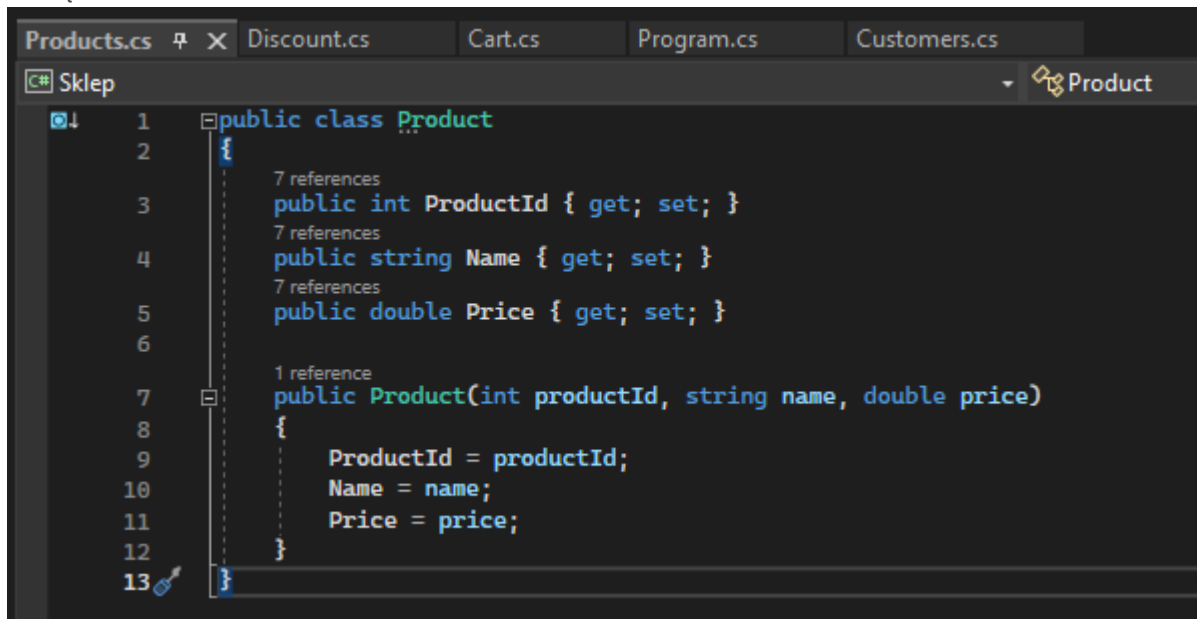
Clarn@clarncc.cc

143075, 143662

14.09.2023

<https://github.com/Clarncc/Sklep>

klasę Product z właściwościami: ProductId, Name, Price



The screenshot shows the Visual Studio IDE with the 'Products.cs' file open. The code defines a 'Product' class with three public properties: 'ProductId' (int), 'Name' (string), and 'Price' (double), each with get and set accessors. A constructor 'Product(int productId, string name, double price)' is also defined, which initializes these properties. The file explorer on the left shows a project named 'Sklep' containing several other files: 'Discount.cs', 'Cart.cs', 'Program.cs', and 'Customers.cs'. The 'Product' class is highlighted in the editor.

```
1 public class Product
2 {
3     7 references
4     public int ProductId { get; set; }
5     7 references
6     public string Name { get; set; }
7     7 references
8     public double Price { get; set; }
9
10    1 reference
11    public Product(int productId, string name, double price)
12    {
13        ProductId = productId;
14        Name = name;
15        Price = price;
16    }
17 }
```

klase Cart

```
3 references
public class Cart
{
    private List<Product> products;
    3 references
    public Customer Customer { get; private set; }

    1 reference
    public Cart(Customer customer)
    {
        Customer = customer;
        products = new List<Product>();
    }

    1 reference
    public void AddToCart(Product product)
    {
        products.Add(product);
    }

    1 reference
    public void ViewCart()
    {
        Console.WriteLine("Cart Contents:");
        foreach (var product in products)
        {
            Console.WriteLine($"Product ID: {product.ProductId}, Name: {product.Name}, Price: {product.Price:C}");
        }
    }

    1 reference
    public double CalculateTotal()
    {
        double total = 0;
        foreach (var product in products)
        {
            total += product.Price;
        }
        return total;
    }

    1 reference
    public List<Product> GetCartContents()
    {
        return products;
    }

    1 reference
    public void LoadCartItems()
    {
        string cartFileName = $"{Customer.CustomerId}_cart.json";
        if (File.Exists(cartFileName))
        {
            string json = File.ReadAllText(cartFileName);
            List<Product> cartItems = JsonConvert.DeserializeObject<List<Product>>(json);
            products = cartItems;
            Console.WriteLine("Cart items loaded from JSON file.");
        }
    }

    1 reference
    public void SaveCartItems()
    {
        string json = JsonConvert.SerializeObject(products);
        string cartFileName = $"{Customer.CustomerId}_cart.json"; // Use a unique file name for each customer's cart
        File.WriteAllText(cartFileName, json);
        Console.WriteLine("Cart items saved to JSON file.");
    }
}
```

klasę `Customer` z właściwościami: `CustomerId`, `Name`, `Cart` oraz metodą `PlaceOrder`.

```
11 references
3 public class Customer
4 {
5     5 references
6     public int CustomerId { get; private set; }
7     8 references
8     public string Name { get; private set; }
9     7 references
10    private Cart Cart { get; set; }
11
12    1 reference
13    public Customer(int customerId, string name)
14    {
15        CustomerId = customerId;
16        Name = name;
17        Cart = new Cart(this);
18    }
19
20    1 reference
21    public void AddToCart(Product product)
22    {
23        Cart.AddToCart(product);
24        SaveCartItems();
25    }
26
27    0 references
28    public void ViewCart()
29    {
30        Cart.ViewCart();
31    }
32
33    1 reference
34    public double CalculateTotal()
35    {
36        return Cart.CalculateTotal();
37    }
38
39    2 references
40    public List<Product> GetCartContents()
41    {
42        return Cart.GetCartContents();
43    }
44
45    2 references
46    public void LoadCartItems()
47    {
48        Cart.LoadCartItems();
49    }
50
51    1 reference
52    public void PlaceOrder()
53    {
54        double total = CalculateTotal();
55
56        if (total > 0)
57        {
58            // Implement order placement logic here (e.g., sending an order to a server or updating a database).
59            // For this example, we'll just display a message.
60
61            Console.WriteLine($"Order placed for {Name} with a total value of {total:C}");
62        }
63        else
64        {
65            Console.WriteLine("Cart is empty. Cannot place an order.");
66        }
67    }
68
69    1 reference
70    private void SaveCartItems()
71    {
72        Cart.SaveCartItems();
73    }
74 }
```

klasę `DiscountProduct`, która dziedziczy po klasie `Product` i dodaje nową właściwość `Discount`.

```
8  {
9      1 reference
    public class DiscountProduct : Product
10  {
11      1 reference
    public double Discount { get; set; }
12
13      0 references
    public DiscountProduct(int productId, string name, double price, double discount)
14      : base(productId, name, price)
15      {
16          Discount = discount;
17      }
18  }
19
20 }
```