Programowanie Obiektowe Zadanie nr. 4

Oleksandr Denysiuk

Email: oleksandrdenysiuk2@gmail.com

Nr. Albumu: 143068

Data: 22.08.2023

Link do repozytorium : https://github.com/M1qtso1/Exercise4.git

Klasa Program

Tworzenie obiektu klasy Shop, obiektu klasy ConsoleUI i wywołanie metody ShowMenu klasy ConsoleUI w metodzie Main programu.

Metoda ShowMenu obiektu ui. Metoda ta wyświetla listę opcji do wyboru przez użytkownika i wykonuje odpowiednie akcje w zależności od wybranej opcji.

```
// The Program class contains the Main method, which is the entry point to the application
0 references
class Program
{
    // The Main method creates an object of the Shop class, an object of the ConsoleUI class and calls the ShowMenu method of the ConsoleUI class
    0 references
    static void Main(string[] args)
    {
        Shop shop = new Shop(); // creating an object of the Shop class
        ConsoleUI ui = new ConsoleUI(shop); // creating an object of the ConsoleUI class with the given argument
        ui.ShowMenu(); // calling ShowMenu method of the ConsoleUI class
    }
}
```

Klasa Customer

Definiowanie klasy Customer, która reprezentuje klienta sklepu z metodą Buy, która pozwala kupować produkt ze sklepu

Klasa Customer reprezentuje klienta sklepu, który ma imię i może kupować produkty ze sklepu. Klasa ta ma jedno pole, jedną właściwość i jedną metodę.

Deklarowane jest prywatne pole name typu string, które przechowuje imię klienta. Pole to jest dostępne tylko wewnątrz klasy Customer i nie może być zmieniane przez inne klasy.

Konstruktor klasy Customer, który przyjmuje jeden argument name i przypisuje go do pola name. Konstruktor ten służy do tworzenia obiektów klasy Customer z podanym imieniem.

```
// The Customer class represents a store customer with a Buy method that allows purchasing a product from the store
4 references
public class Customer
{
    // The class field is private and accessible only through the get and set method
    private string name; // customer name

    // The Customer class constructor takes one argument and assigns it to the class field
    1 reference
    public Customer(string name)
    {
        this.name = name;
    }

    // The get and set method allows reading and changing the value of the class field
    1 reference
    public string Name
    {
        get { return name; }
        set { name = value; }
    }
```

Metoda Buy, która przyjmuje dwa argumenty: shop i product. Metoda ta pozwala klientowi kupić produkt ze sklepu, sprawdzając, czy jest to zwykły produkt czy produkt ze zniżką i stosując odpowiednią cenę.

Operator is używany jest do sprawdzenia, czy produkt jest typu DiscountProduct, czyli czy ma zniżkę. Jeśli tak, to wykonuje się blok kodu w nawiasach klamrowych po słowie kluczowym if.

Operator rzutowania (DiscountProduct) do przypisania produktu do zmiennej dp typu DiscountProduct. Oznacza to, że zmienna dp ma dostęp do wszystkich pól i metod klasy DiscountProduct, w tym właściwości Discount.

```
// The Buy method allows the customer to purchase a product from the store, checking whether it is a regular product or a product with a discount and applying the appropriate price
1 reference
public void Buy(Shop shop, Product product)
{
    if (product is DiscountProduct) // if the product is of type DiscountProduct, calculates the price after discount and subtracts it from the customer's balance
    {
        DiscountProduct dp = (DiscountProduct)product; // casting type Product to type DiscountProduct
        decimal priceAfterDiscount = dp.Price * (1 - dp.Discount); // calculating the price after discount
        Console.WriteLine($"Customer {name} bought product {dp.Name} from store {shop} for {priceAfterDiscount} USD.\n____________________");
    }
    else // if the product is of type Product, uses the base price and subtracts it from the customer's balance
    {
        Console.WriteLine($"Customer {name} bought product {product.Name} from store {shop} for {product.Price} USD.\n____________________");
    }
}
```

Klasa DiscountProduct

Definiowanie klasy DiscountProduct, która dziedziczy po klasie Product i dodaje nową właściwość Discount

Klasa DiscountProduct reprezentuje produkt ze zniżką, który ma nazwę, cenę, ilość i procent zniżki. Klasa ta jest podklasą klasy Product, która reprezentuje ogólny produkt

Prywatne pole discount typu decimal, które przechowuje wartość zniżki produktu. Pole to jest dostępne tylko wewnątrz klasy DiscountProduct i nie może być zmieniane przez inne klasy.

Definiowany jest konstruktor klasy DiscountProduct, który przyjmuje cztery argumenty: name, price, quantity i discount. Konstruktor ten wywołuje konstruktor klasy bazowej Product za pomocą base przekazuje mu trzy pierwsze argumenty. Następnie konstruktor ten przypisuje wartość czwartego argumentu do pola discount.

Metoda ToString klasy Product zwraca tekstową reprezentację produktu ze zniżką, która zawiera informacje o nazwie, cenie, ilości i procentowej wartości zniżki. Metoda ta używa metody ToString klasy bazowej za pomocą słowa kluczowego base i dodaje do niej informację o zniżce.

```
{// The DiscountProduct class inherits from the Product class and adds a new property Discount
    6 references
    public class DiscountProduct : Product
    {
        private decimal discount; // product discount

        // The DiscountProduct class constructor takes four arguments and passes three of them to the base class Product constructor
        1 reference
        public DiscountProduct(string name, decimal price, int quantity, decimal discount) : base(name, price, quantity)
        {
            this.discount = discount;
        }
        // The get and set method allows reading and changing the value of the class field
        1 reference
        public decimal Discount
        {
            get { return discount; }
            set { discount = value; }
        }

        // The ToString method returns a text representation of the product with a discount
        2 references
        public override string ToString()
        {
            return $"{base.ToString()} - {discount * 100}% off";
        }
    }
```

Klasa Product

Definiowanie klasy Product, która reprezentuje ogólny produkt ze sklepu.

Prywatne pola name, price i quantity typów string, decimal i int, które przechowują informacje o produkcie. Pola te są dostępne tylko wewnątrz klasy Product i nie mogą być zmieniane przez inne klasy.

Konstruktor klasy Product, który przyjmuje trzy argumenty: name, price i quantity i przypisuje je do pól name, price i quantity. Konstruktor ten służy do tworzenia obiektów klasy Product z podanymi parametrami.

```
public class Product
{
    private string name; // product name
    private decimal price; // product price
    private int quantity; // product quantity

    // The Product class constructor takes three arguments and assigns them to the class fields
    2 references
    public Product(string name, decimal price, int quantity)
    {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    // The get and set methods allow reading and changing the values of the class fields
    5 references
    public string Name
    {
        get { return name; }
        set { name = value; }
    }

    3 references
    public decimal Price
    {
        get { return price; }
        set { price = value; }
    }
```

```
    2 references
    public int Quantity
    {
        get { return quantity; }
        set { quantity = value; }
    }
```

Metoda ToString klasy Object zwraca tekstową reprezentację produktu, która zawiera informacje o nazwie, cenie i ilości produktu.

```
    // The ToString method returns a text representation of the product
    2 references
    public override string ToString()
    {
        return $"{name} - {price} USD - {quantity} item(s)\n____________________";
    }
```

Klasa ConsoleUI

Definiowana jest klasa ConsoleUI, która reprezentuje interfejs użytkownika oparty na konsoli.

Prywatne pole shop typu Shop, które przechowuje sklep, który współpracuje z interfejsem użytkownika.

Konstruktor klasy ConsoleUI, który przyjmuje jeden argument typu Shop i przypisuje go do pola shop. Konstruktor ten służy do tworzenia obiektów klasy ConsoleUI z podanym sklepem.

```
// The ConsoleUI class represents a user interface based on a console
3 references
public class ConsoleUI
{
    // The class field is private and accessible only through the get and set method
    private Shop shop; // store that cooperates with the user interface

    // The ConsoleUI class constructor takes one argument and assigns it to the class field
    1 reference
    public ConsoleUI(Shop shop)
    {
        this.shop = shop;
    }

    // The get and set method allows reading and changing the value of the class field
    0 references
    public Shop Shop
    {
        get { return shop; }
        set { shop = value; }
    }
```

Metoda ShowMenu, która wyświetla główne menu aplikacji i pozwala użytkownikowi wybrać opcję. Metoda ta używa zmiennej exit do kontrolowania głównej pętli aplikacji, która działa dopóki użytkownik nie wybierze opcji wyjścia. Metoda ta używa też metody WriteLine do wyświetlania tekstu i metody ReadLine do odczytywania wyboru użytkownika. Metoda ta używa też instrukcji switch do wykonania odpowiedniej akcji w zależności od wyboru użytkownika, wywołując inne metody klasy ConsoleUI lub wyświetlając komunikat o błędzie.

```
// The ShowMenu method displays the main menu of the application and allows the user to choose an option
1 reference
public void ShowMenu()
{
    bool exit = false; // variable controlling the main loop of the application
    while (!exit) // main loop of the application runs until user chooses exit option
    {
        Console.WriteLine("Oleksandr Denysiuk's shop =)\n_____________________");
        Console.WriteLine("Choose an option:");
        Console.WriteLine("1. Add a product to the store");
        Console.WriteLine("2. Display a list of products in the store");
        Console.WriteLine("3. Sell a product to a customer");
        Console.WriteLine("4. Save a list of products to a file");
        Console.WriteLine("5. Load a list of products from a file");
        Console.WriteLine("0. Exit from application\n_____________________");
        string choice = Console.ReadLine(); // reading user's choice
```

```
switch (choice) // performing appropriate action depending on user's choice
{
    case "1":
        AddProduct(); // calling AddProduct method
        break;
    case "2":
        ShowProducts(); // calling ShowProducts method
        break;
    case "3":
        SellProduct(); // calling SellProduct method
        break;
    case "4":
        SaveProducts(); // calling SaveProducts method
        break;
    case "5":
        LoadProducts(); // calling LoadProducts method
        break;
    case "0":
        exit = true; // setting exit variable to true, which will end main loop of application
        break;
    default:
        Console.WriteLine("Invalid choice. Try again.\n_____________________"); // displaying error message if user enters incorrect option
        break;
}
```

Metoda AddProduct, która pozwala użytkownikowi dodać produkt do sklepu, pytając o jego właściwości i tworząc odpowiedni obiekt klasy Product lub DiscountProduct. Metoda ta używa metody ReadLine do odczytywania danych od użytkownika i pętli while do sprawdzania, czy dane nie są puste. Metoda ta używa też instrukcji warunkowej if do sprawdzania, czy produkt ma zniżkę. Jeśli tak, to tworzy obiekt klasy DiscountProduct za pomocą operatora new i konstruktora z czterema argumentami. Jeśli nie, to tworzy obiekt klasy Product za pomocą operatora new i konstruktora z trzema argumentami. Następnie metoda ta dodaje obiekt do listy produktów w sklepie za pomocą metody AddProduct klasy Shop.

```
// The AddProduct method allows user to add a product to store, asking for its properties and creating appropriate object of Product or DiscountProduct class
1 reference
public void AddProduct()
{
    Console.WriteLine("Adding product to store.");
    Console.Write("Enter product name: ");
    string name = Console.ReadLine(); // reading product name from user
    while (string.IsNullOrEmpty(name)) // while the input is empty
    {
        Console.WriteLine("You did not enter anything. Please try again.");
        Console.Write("Enter product name: ");
        name = Console.ReadLine(); // reading product name from user again
    }
```

Zmienna resultPrice typu bool przechowuje wynik próby konwersji ciągu znaków na liczbę za pomocą metody TryParse klasy decimal. Metoda ta przyjmuje dwa argumenty: inputPrice i price. Pierwszy argument to ciąg znaków do konwersji, a drugi argument to zmienna, do której zostanie przypisana skonwertowana wartość. Metoda ta zwraca wartość true, jeśli konwersja się powiedzie, lub false, jeśli się nie powiedzie.

Pętla while z warunkiem !resultPrice, czyli pętla będzie się wykonywać dopóki wynik konwersji będzie false, czyli dopóki użytkownik nie wpisze poprawnej liczby. Ponownie próbowana jest konwersja ciągu znaków na liczbę za pomocą metody TryParse klasy decimal. Wynik tej metody

jest przypisywany do zmiennej resultPrice typu bool ponownie. Jeśli konwersja się powiedzie, to pętla while się zakończy i program przejdzie do następnego kroku. Jeśli nie, to pętla while będzie się powtarzać, dopóki użytkownik nie wpisze poprawnej liczby.

```
    // if the input is not empty, the loop will end and program will go for the next step
    Console.Write("Enter product price: ");
    string inputPrice = Console.ReadLine(); // reading product price from user as a string
    decimal price;
    bool resultPrice = decimal.TryParse(inputPrice, out price); // trying to parse the input to a decimal type
    while (!resultPrice) // while the parsing failed
    {
        Console.WriteLine("You did not enter a valid number. Please try again.");
        Console.Write("Enter product price: ");
        inputPrice = Console.ReadLine(); // reading product price from user as a string again
        resultPrice = decimal.TryParse(inputPrice, out price); // trying to parse the input to a decimal type again
    }
    // if the parsing was successful, the loop will end and program will go for the next step
```

```
Console.Write("Enter product quantity: ");
string inputQuantity = Console.ReadLine(); // reading product quantity from user as a string
int quantity;
bool resultQuantity = int.TryParse(inputQuantity, out quantity); // trying to parse the input to an int type
while (!resultQuantity) // while the parsing failed
{
    Console.WriteLine("You did not enter a valid number. Please try again.");
    Console.Write("Enter product quantity: ");
    inputQuantity = Console.ReadLine(); // reading product quantity from user as a string again
    resultQuantity = int.TryParse(inputQuantity, out quantity); // trying to parse the input to an int type again
}
// if the parsing was successful, the loop will end and program will go for the next step
```

Instrukcja warunkowa if używana jest z warunkiem answer == “Y” || answer == “y”, czyli jeśli odpowiedź jest Y lub y, to oznacza, że produkt ma zniżkę i należy utworzyć obiekt klasy DiscountProduct. Jeśli tak, to wykonuje się blok kodu w nawiasach klamrowych po słowie kluczowym if.

```
Console.Write("If product has discount, type 'Y', if not, leave the field empty: ");
string answer = Console.ReadLine(); // reading answer from user, whether product has discount
if (answer == "Y" || answer == "y") // if answer is Y or y, it means that product has discount and object of DiscountProduct class should be created
{
    Console.Write("Enter discount amount (e.g. 0,1 means 10%): ");
    string inputDiscount = Console.ReadLine(); // reading product discount from user as a string
    decimal discount;
    bool resultDiscount = decimal.TryParse(inputDiscount, out discount); // trying to parse the input to a decimal type
    while (!resultDiscount) // while the parsing failed
    {
        Console.WriteLine("You did not enter a valid number. Please try again.");
        Console.Write("Enter discount amount (e.g. 0,1 means 10%): ");
        inputDiscount = Console.ReadLine(); // reading product discount from user as a string again
        resultDiscount = decimal.TryParse(inputDiscount, out discount); // trying to parse the input to a decimal type again
    }
    // if the parsing was successful, the loop will end and program will go for the next step
    DiscountProduct dp = new DiscountProduct(name, price, quantity, discount); // creating object of DiscountProduct class with given properties
    shop.AddProduct(dp); // adding object to list of products in store
}
else // if answer is not Y or y, it means that product does not have discount and object of Product class should be created
{
    Product p = new Product(name, price, quantity); // creating object of Product class with given properties
    shop.AddProduct(p); // adding object to list of products in store
}
Console.WriteLine("Product has been added to store.\n____________________"); // displaying information about success
```

Metoda ShowProducts, która wywołuje metodę ShowProducts klasy Shop, która wyświetla informacje o wszystkich produktach w sklepie.

```
// The ShowProducts method calls the ShowProducts method of the Shop class, which displays information about all products in the store
1 reference
public void ShowProducts()
{
    Console.WriteLine("Displaying list of products in store.");
    shop.ShowProducts(); // calling ShowProducts method of the Shop class
}
// The SellProduct method allows the user to sell a product to a customer, asking for the product name and customer name and calling the SellProduct method of the Shop class
```

Metoda SellProduct, która pozwala użytkownikowi sprzedać produkt klientowi, pytając o nazwę produktu i nazwę klienta i wywołując metodę SellProduct klasy Shop. Metoda ta używa metody ReadLine do odczytywania danych od użytkownika i metody FindProductByName do wyszukiwania produktu o podanej nazwie na liście produktów w sklepie. Metoda ta używa też instrukcji warunkowej if i operatora != do sprawdzania, czy produkt o podanej nazwie został znaleziony. Jeśli tak, to tworzy obiekt klasy Customer z podaną nazwą i wywołuje metodę SellProduct pola shop z podanymi argumentami. Jeśli nie, to wyświetla wiadomość o błędzie.

```
// The SellProduct method allows the user to sell a product to a customer, asking for the product name and customer name and calling the SellProduct method of the Shop class
1 reference
public void SellProduct()
{
    Console.WriteLine("Selling product to customer.");
    Console.Write("Enter product name: ");
    string name = Console.ReadLine(); // reading product name from user
    Product product = FindProductByName(name); // finding product with given name on list of products in store using FindProductByName method
    if (product != null) // if product with given name was found
    {
        Console.Write("Enter customer name: ");
        string customerName = Console.ReadLine(); // reading customer name from user
        Customer customer = new Customer(customerName); // creating object of Customer class with given name
        shop.SellProduct(product, customer); // calling SellProduct method of Shop class with given arguments
    }
    else // if product with given name was not found, displaying error message
    {
        Console.WriteLine($"There is no such product in store.\n{new string('_', 50)}");
    }
}
```

Metoda SaveProducts, która pozwala użytkownikowi zapisać listę produktów do pliku JSON, pytając o nazwę pliku i wywołując metodę SaveProducts klasy Shop. Metoda ta używa metody ReadLine do odczytywania nazwy pliku od użytkownika i operatora . do wywołania metody SaveProducts pola shop z podanym argumentem.

```
// The SaveProducts method allows the user to save the list of products to a JSON file, asking for the file name and calling the SaveProducts method of the Shop class
1 reference
public void SaveProducts()
{
    Console.WriteLine("Saving list of products to a file.");
    Console.Write("Enter file name: ");
    string fileName = Console.ReadLine(); // reading file name from user
    shop.SaveProducts(fileName); // calling SaveProducts method of Shop class with given argument
    Console.WriteLine("List of products has been saved to a file.\n{new string('_', 50)}"); // displaying information about success
}
```

Metoda LoadProducts, która pozwala użytkownikowi załadować listę produktów z pliku JSON, pytając o nazwę pliku i wywołując metodę LoadProducts klasy Shop. Metoda ta używa metody ReadLine do odczytywania nazwy pliku od użytkownika i operatora . do wywołania metody LoadProducts pola shop z podanym argumentem.

```
// The LoadProducts method allows the user to load the list of products from a JSON file, asking for the file name and calling the LoadProducts method of the Shop class
1 reference
public void LoadProducts()
{
    Console.WriteLine("Loading list of products from a file.");
    Console.Write("Enter file name: ");
    string fileName = Console.ReadLine(); // reading file name from user
    shop.LoadProducts(fileName); // calling LoadProducts method of Shop class with given argument
    Console.WriteLine("List of products has been loaded from a file.\n{new string('_', 50)}"); // displaying information about success
}
```

Metoda FindProductByName, która wyszukuje produkt o podanej nazwie na liście produktów w sklepie i zwraca go lub null, jeśli nie zostanie znaleziony. Metoda ta przyjmuje jeden argument typu string i używa słowa kluczowego foreach do iterowania po liście produktów w sklepie za pomocą właściwości Products pola shop. Metoda ta używa też instrukcji warunkowej if i operatora == do porównywania nazwy produktu z podaną nazwą. Jeśli są równe, to zwraca produkt za pomocą operatora return. Jeśli nie są równe, to kontynuuje pętlę. Jeśli żaden produkt nie pasuje do podanej nazwy, to zwraca null.

```
// The FindProductByName method searches for a product with a given name on the list of products in the store and returns it or null if not found
1 reference
public Product FindProductByName(string name)
{
    foreach (Product product in shop.Products) // going through the list of products in the store
    {
        if (product.Name == name) // if the product name is equal to the given name
        {
            return product; // returns the product
        }
    }
    return null; // if no product with the given name was found, returns null
}
```

Klasa Shop

Definiowanie klasy Shop, która reprezentuje sklep z listą produktów i metodami do zarządzania nimi.

Klasa Shop reprezentuje sklep, który ma listę produktów i może dodawać, wyświetlać, sprzedawać i zapisywać produkty.

Prywatne pole products typu List<Product>, które przechowuje listę produktów w sklepie. Pole to jest dostępne tylko wewnątrz klasy Shop i nie może być zmieniane przez inne klasy.

Konstruktor klasy Shop tworzy pustą listę produktów za pomocą operatora new i klasy List<Product>. Konstruktor ten służy do tworzenia obiektów klasy Shop bez podawania żadnych parametrów.

```
// The Shop class represents a store with a list of products and methods for managing them
7 references
public class Shop
{
    private List<Product> products; // list of products

    // The Shop class constructor creates an empty list of products
    1 reference
    public Shop()
    {
        products = new List<Product>();
    }

    // The get method returns the list of products
    1 reference
    public List<Product> Products
    {
        get { return products; }
    }
```

Metoda AddProduct przyjmuje jeden argument typu Product i dodaje go do listy produktów za pomocą metody Add klasy List<Product>. Metoda ta służy do dodawania nowych produktów do sklepu.

```
    // The AddProduct method adds a product to the list of products
    2 references
    public void AddProduct(Product product)
    {
        products.Add(product);
    }
```

Metoda SellProduct, która przyjmuje dwa argumenty: product typu Product i customer typu Customer i sprzedaje produkt klientowi, zmniejszając jego ilość w sklepie i wyświetlając informacje o sprzedaży. Metoda ta sprawdza, czy produkt jest dostępny za pomocą instrukcji warunkowej if i operatora >. Jeśli tak, to zmniejsza ilość produktu o 1 za pomocą operatora --, wywołuje metodę Buy klienta za pomocą operatora, a następnie wyświetla informacje o sprzedaży za pomocą metody WriteLine. Jeśli nie, to wyświetla wiadomość o braku produktu na

stanie.

```
// The ShowProducts method displays information about all products in the store
1 reference
public void ShowProducts()
{
    Console.WriteLine("List of products in the store: \n");
    foreach (Product product in products)
    {
        Console.WriteLine(product + "\n_____________________");
    }
}

// The SellProduct method sells a product to a customer, reduces its quantity in the store and displays information about the sale
1 reference
public void SellProduct(Product product, Customer customer)
{
    if (product.Quantity > 0) // checks if the product is available
    {
        product.Quantity--; // reduces the product quantity by 1
        customer.Buy(this, product); // calls the Buy method of the Customer
        Console.WriteLine($"Sold product {product.Name} to customer {customer.Name} for {product.Price} USD.\n_____________________");
    }
    else // if the product is unavailable, displays a message about out of stock
    {
        Console.WriteLine($"Product {product.Name} is out of stock.\n_____________________");
    }
}
```

Metoda SaveProducts pozwala użytkownikowi zapisać listę produktów do pliku JSON, pytając o nazwę pliku i używając narzędzi do serializacji danych. Metoda ta tworzy opcje serializacji z ignorowaniem wartości domyślnych i formatowaniem wcięć za pomocą klasy JsonSerializerOptions. Następnie sprawdza, czy nazwa pliku ma rozszerzenie .json za pomocą metody GetExtension klasy Path i pętli while. Jeśli nie, to prosi użytkownika o podanie poprawnej nazwy pliku. Jeśli tak, to tworzy strumień plikowy do zapisywania danych za pomocą klasy FileStream i metody Create. Następnie serializuje listę produktów do formatu JSON za pomocą klasy JsonSerializer i metody SerializeAsync.

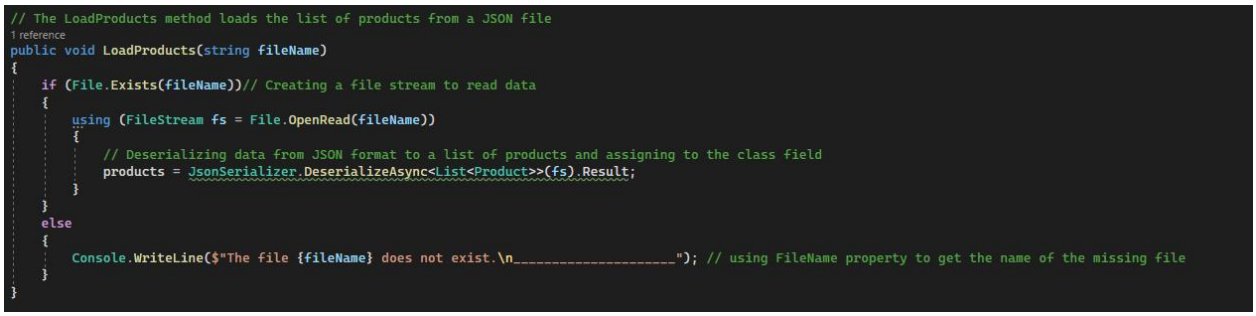
```
// The SaveProducts method allows the user to save the list of products to a JSON file, asking for the file name and calling the SaveProducts method of the Shop class
1 reference
public void SaveProducts(string fileName)
{
    // Creating serialization options with ignoring default values and formatting indents
    var options = new JsonSerializerOptions
    {
        IgnoreNullValues = true,
        WriteIndented = true
    };

    // Checking if the file name has a .json extension
    string extension = Path.GetExtension(fileName);
    while (extension != ".json") // while the extension is not .json
    {
        Console.WriteLine("The file name is invalid. Please enter a file name with a .json extension.");
        Console.Write("Enter file name: ");
        fileName = Console.ReadLine(); // reading file name from user again
        extension = Path.GetExtension(fileName); // getting the extension again
    }
    // If the extension is .json, the loop will end and you can continue with your code

    // Creating a file stream to write data
    using (FileStream fs = File.Create(fileName))
    {
        // Serializing the list of products to JSON format and writing to the file stream
        JsonSerializer.SerializeAsync(fs, products, options);
    }
}
```

Metoda LoadProducts ładuje listę produktów z pliku JSON. Metoda ta sprawdza, czy plik istnieje za pomocą metody Exists klasy File. Jeśli tak, to tworzy strumień plikowy do odczytywania

danych za pomocą klasy FileStream i metody OpenRead. Następnie deserializuje dane z formatu JSON do listy produktów za pomocą klasy JsonSerializer i metody DeserializeAsync i przypisuje je do pola products. Jeśli nie, to wyświetla wiadomość o braku pliku.



```
// The LoadProducts method loads the list of products from a JSON file
1 reference
public void LoadProducts(string fileName)
{
    if (File.Exists(fileName))// Creating a file stream to read data
    {
        using (FileStream fs = File.OpenRead(fileName))
        {
            // Deserializing data from JSON format to a list of products and assigning to the class field
            products = JsonSerializer.DeserializeAsync<List<Product>>(fs).Result;
        }
    }
    else
    {
        Console.WriteLine($"The file {fileName} does not exist.\n_____________________"); // using FileName property to get the name of the missing file
    }
}
```