



**WYŻSZA SZKOŁA
INFORMATYKI i ZARZĄDZANIA**
z siedzibą w Rzeszowie

KOLEGIUM INFORMATYKI STOSOWANEJ

Kierunek: INFORMATYKA

Specjalność: PROGRAMOWANIE

Patrycja Socha
Nr albumu studenta: 71464

System zarządzania sklepem odzieżowym

Prowadzący: mgr inż. Adrian Hałys

PROJEKT ZALICZENIOWY

Rzeszów 2026

Spis treści

Wstęp	4
1 Charakterystyka danych	5
1.1 Atrybuty produktu	5
2 Instrukcja obsługi programu	7
2.1 Menu główne	7
2.2 Scenariusze użycia	8
2.2.1 Dodawanie towaru	8
2.2.2 Edycja i Usuwanie	8
3 Implementacja techniczna	9
3.1 Zarządzanie danymi	9
3.2 Trwałość danych	9
4 Link do repozytorium z plikami do projektu	11

Wstęp

Celem niniejszego projektu było zaprojektowanie i zaimplementowanie systemu informatycznego wspomagającego zarządzanie asortymentem w sklepie odzieżowym. Aplikacja została stworzona w języku C# w środowisku .NET, zgodnie z paradigmatem programowania obiektowego.

Głównym zadaniem programu jest umożliwienie pracownikowi sklepu wykonywania podstawowych operacji na danych (CRUD – Create, Read, Update, Delete), takich jak dodawanie nowych produktów, przeglądanie stanu magazynowego, edycja parametrów odzieży oraz usuwanie wycofanych produktów.

W projekcie położono nacisk na trwałość danych poprzez zastosowanie mechanizmu serializacji do formatu JSON, co pozwala na zachowanie stanu magazynu pomiędzy uruchomieniami aplikacji. Interfejs użytkownika został zrealizowany w formie aplikacji konsolowej, co zapewnia szybkość działania i przejrzystość obsługi.

Rozdział 1

Charakterystyka danych

Podstawowym elementem systemu jest model reprezentujący pojedynczy produkt odzieżowy. W celu odwzorowania rzeczywistych cech asortymentu, zdefiniowano klasę `ProduktOdziezowy`, która przechowuje kluczowe atrybuty każdego towaru.

1.1 Atrybuty produktu

Każdy obiekt w systemie opisany jest następującym zestawem cech (właściwości):

- **Id** (liczba całkowita) – unikalny identyfikator produktu. Jest on niezbędny do jednoznacznego wskazania towaru, który chcemy edytować lub usunąć, nawet jeśli inne cechy (np. nazwa) są takie same.
- **Nazwa** (tekst) – określenie typu ubioru, np. „Koszulka Polo”.
- **Marka** (tekst) – producent odzieży.
- **Rozmiar** (tekst) – parametr określający wielkość, np. „L”, „42”, „XL”. Zastosowano typ tekstowy, aby obsłużyć różne standardy rozmiarów.
- **Kolor** (tekst) – barwa produktu.
- **Cena** (liczba zmiennoprzecinkowa) – wartość produktu wyrażona w PLN.

The screenshot shows a code editor with three tabs at the top: 'ProduktOdziezowy.cs', 'Program.cs', and 'SklepManager.cs'. The 'ProduktOdziezowy.cs' tab is active, displaying the following C# code:

```
Projekt_PO_Patrycja_Socha_71464 SklepOdziezowy.ProduktOdziezowy

1 using System;
2
3 namespace SklepOdziezowy
4 {
5     public class ProduktOdziezowy
6     {
7         public int Id { get; set; }
8         public string Nazwa { get; set; }
9         public string Marka { get; set; }
10        public string Rozmiar { get; set; }
11        public string Kolor { get; set; }
12        public double Cena { get; set; }
13    }
14}
```

The code defines a class `ProduktOdziezowy` with properties `Id`, `Nazwa`, `Marka`, `Rozmiar`, `Kolor`, and `Cena`. The `Id` property is of type `int` and has `get` and `set` accessors. The `Nazwa`, `Marka`, `Rozmiar`, `Kolor`, and `Cena` properties are of type `string` and also have `get` and `set` accessors. The `Cena` property is of type `double` and has `get` and `set` accessors.

Powyższa struktura danych pozwala na elastyczne zarządzanie różnymi typami odzieży bez konieczności modyfikacji kodu źródłowego programu. Klasa posiada również przeciążoną metodę `ToString()`, która odpowiada za czytelne wyświetlanie informacji o produkcie w konsoli.

Rozdział 2

Instrukcja obsługi programu

Interfejs aplikacji został zaprojektowany w sposób intuicyjny. Jest to menu tekstowe, po którym użytkownik nawiguje wpisując odpowiednie cyfry na klawiaturze.

2.1 Menu główne

Po uruchomieniu programu użytkownikowi prezentowane jest menu zawierające opcje:

1. **Wyświetl wszystkie produkty** – prezentuje listę całego asortymentu.
2. **Dodaj nowy produkt** – uruchamia procedurę wprowadzania towaru.
3. **Edytuj produkt** – pozwala zmienić dane istniejącego towaru.
4. **Usuń produkt** – trwale usuwa towar z magazynu.
5. **Wyjście** – bezpiecznie zamyka aplikację.



```
ProduktOdziewowy.cs  X  Program.cs  X  SklepManager.cs
Projekt_PO_Patrycja_Socha_71464
Odwołania: 0
Odwołania: 0
class Program
{
    Odwołania: 0
    static void Main(string[] args)
    {
        SklepManager manager = new SklepManager();
        while (true)
        {
            Console.Clear();
            Console.WriteLine("== SYSTEM ZARZĄDZANIA SKLEPEM ODZIEŻOWYM ==");
            Console.WriteLine("1. Wyświetl produkty");
            Console.WriteLine("2. Dodaj produkt");
            Console.WriteLine("3. Edytuj produkt");
            Console.WriteLine("4. Usuń produkt");
            Console.WriteLine("0. Wyjście");
            Console.Write("\nWybierz opcję: ");
            string wybor = Console.ReadLine();
            switch (wybor)
            {
                case "1":
                    WyświetlProdukty(manager);
                    break;
                case "2":
                    DodaNowyProdukt(manager);
                    break;
                case "3":
                    EdytujProduktInterfejs(manager);
                    break;
                case "4":
                    UsunProduktInterfejs(manager);
                    break;
                case "0":
                    return;
                default:
                    Console.WriteLine("Nieznana opcja.");
                    break;
            }
        }
    }
}
```

2.2 Scenariusze użycia

2.2.1 Dodawanie towaru

Po wybraniu opcji nr 2, system prosi o podanie kolejnych atrybutów. Program posiada wbudowaną walidację – w przypadku wprowadzenia błędnej ceny (np. liter zamiast cyfr), system wyświetli komunikat o błędzie (blok `try-catch`) i nie pozwoli na wprowadzenie uszkodzonych danych.

```
1 odwołanie
static void DodajNowyProdukt(SklepManager manager)
{
    Console.WriteLine("\n--- DODAWANIE ---");
    try
    {
        Console.Write("ID: "); int id = int.Parse(Console.ReadLine());
        Console.Write("Nazwa: "); string nazwa = Console.ReadLine();
        Console.Write("Marka: "); string marka = Console.ReadLine();
        Console.Write("Rozmiar: "); string rozmiar = Console.ReadLine();
        Console.Write("Kolor: "); string kolor = Console.ReadLine();
        Console.Write("Cena: "); double cena = double.Parse(Console.ReadLine());

        manager.DodajProdukt(new ProduktOdziezowy(id, nazwa, marka, rozmiar, kolor, cena));
        Console.WriteLine("Dodano!");
    }
    catch { Console.WriteLine("Błąd danych!"); }
    Console.ReadLine();
}
```

2.2.2 Edycja i Usuwanie

Aby zmienić dane lub usunąć produkt, użytkownik musi podać jego numer ID. System najpierw sprawdza, czy produkt o takim numerze istnieje w bazie. Jeśli tak – wykonuje operację i automatycznie zapisuje zmiany w pliku.

```
76     1 odwołanie
77     static void EdytujProduktInterfejs(SklepManager manager)
78     {
79         Console.WriteLine("\n--- EDYCJA ---");
80         Console.Write("Podaj ID: ");
81         if (int.TryParse(Console.ReadLine(), out int id))
82         {
83             Console.Write("Nowa nazwa: "); string nazwa = Console.ReadLine();
84             Console.Write("Nowa cena: ");
85             if (double.TryParse(Console.ReadLine(), out double cena))
86             {
87                 manager.EdytujProdukt(id, nazwa, cena);
88                 Console.WriteLine("Zmieniono.");
89             }
90         }
91         Console.ReadLine();
92     }
93
94     1 odwołanie
95     static void UsunProduktInterfejs(SklepManager manager)
96     {
97         Console.WriteLine("\n--- USUWANIE ---");
98         Console.Write("Podaj ID: ");
99         if (int.TryParse(Console.ReadLine(), out int id)) manager.UsunProdukt(id);
100    }
```

Rozdział 3

Implementacja techniczna

System został zrealizowany w języku C# przy użyciu platformy .NET. Projekt podzielono na warstwy, oddzielając interfejs użytkownika (plik `Program.cs`) od logiki biznesowej (plik `SklepManager.cs`).

3.1 Zarządzanie danymi

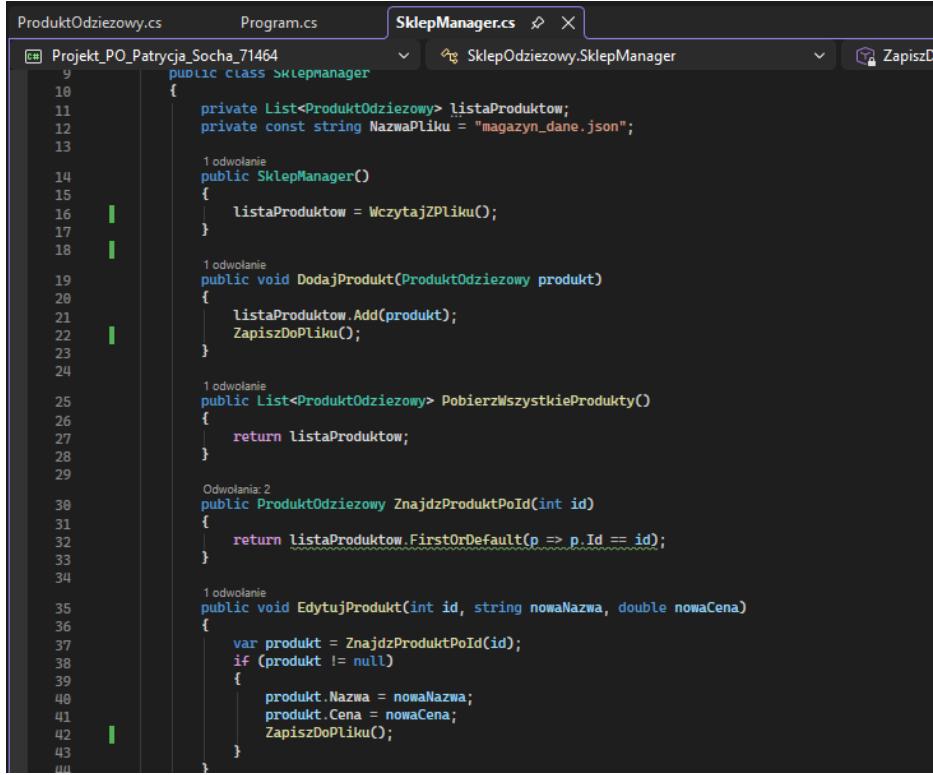
Kluczowym elementem systemu jest klasa `SklepManager`. Przechowuje ona listę obiektów w pamięci RAM, wykorzystując generyczną kolekcję `List<ProduktDziezowy>`. Dzięki wykorzystaniu biblioteki `System.Linq`, operacje wyszukiwania produktów (np. po ID) są realizowane wydajnie za pomocą metody `FirstOrDefault`.

3.2 Trwałość danych

Aby spełnić wymagania projektowe dotyczące zapisu danych, wykorzystano format **JSON** (JavaScript Object Notation). Wybór ten podyktowany był czytelnością formatu oraz łatwością obsługi w języku C#.

Proces zapisu danych przebiega następująco:

1. Po każdej operacji zmieniającej stan magazynu (dodanie, edycja, usunięcie), wywoływana jest metoda `ZapiszDoPliku()`.
2. Lista obiektów jest poddawana **serializacji** do formatu tekstowego JSON za pomocą biblioteki `System.Text.Json`.
3. Powstały tekst jest zapisywany w pliku `magazyn_dane.json` na dysku twardym.



```
ProduktOdziezowy.cs      Program.cs      SklepManager.cs      ZapisZPliku.cs
Projekt_PO_Patrycja_Socha_71464
  9  public class SklepManager
10 {
11     private List<ProduktOdziezowy> listaProduktow;
12     private const string NazwaPliku = "magazyn_dane.json";
13
14     1 odwołanie
15     public SklepManager()
16     {
17         listaProduktow = WczytajZPliku();
18     }
19
20     1 odwołanie
21     public void DodajProdukt(ProduktOdziezowy produkt)
22     {
23         listaProduktow.Add(produkt);
24         ZapiszDoPliku();
25     }
26
27     1 odwołanie
28     public List<ProduktOdziezowy> PobierzWszystkieProdukty()
29     {
30         return listaProduktow;
31     }
32
33     Odwołania: 2
34     public ProduktOdziezowy ZnajdzProduktPoId(int id)
35     {
36         return listaProduktow.FirstOrDefault(p => p.Id == id);
37     }
38
39     1 odwołanie
40     public void EdytujProdukt(int id, string nowaNazwa, double nowaCena)
41     {
42         var produkt = ZnajdzProduktPoId(id);
43         if (produkt != null)
44         {
45             produkt.Nazwa = nowaNazwa;
46             produkt.Cena = nowaCena;
47             ZapiszDoPliku();
48         }
49     }
50 }
```

Przy uruchamianiu programu następuje proces odwrotny (deserializacja), który odtwarza listę obiektów. Program jest zabezpieczony na wypadek braku pliku (np. przy pierwszym uruchomieniu) – w takiej sytuacji tworzona jest nowa, pusta baza danych.

Rozdział 4

Link do repozytorium z plikami do projektu

Kliknij tutaj, aby przejść do plików