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

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 paradygmatem 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

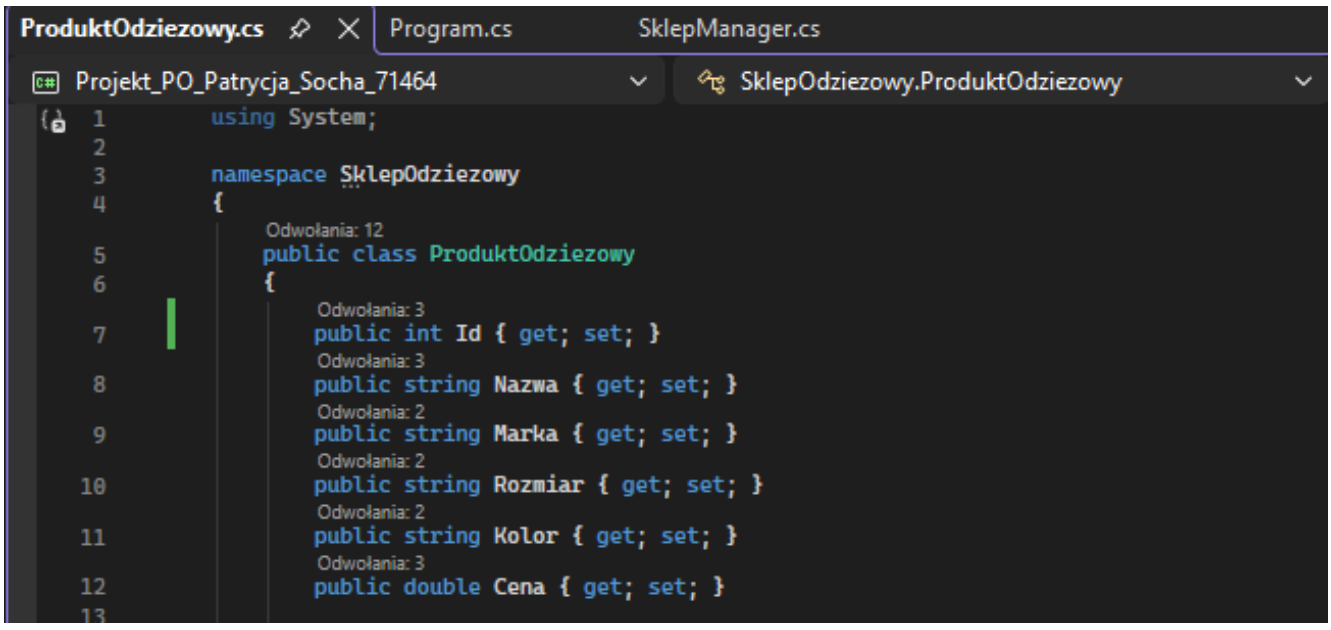
Charakterystyka danych

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 rozmiarówki.
- **Kolor** (tekst) – barwa produktu.
- **Cena** (liczba zmiennoprzecinkowa) – wartość produktu wyrażona w PLN.



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.

ProduktOdziezowy.cs | Program.cs | SklepManager.cs

Projekt_PO_Patrycja_Socha_71464 | Program | Main(string[] args)

```
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":
                    WyswietlProdukty(manager);
                    break;
                case "2":
                    DodajNowyProdukt(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.

```
1 odwołanie
static void EdytujProduktInterfejs(SklepManager manager)
{
    Console.WriteLine("\n--- EDYCJA ---");
    Console.Write("Podaj ID: ");
    if (int.TryParse(Console.ReadLine(), out int id))
    {
        Console.Write("Nowa nazwa: "); string nazwa = Console.ReadLine();
        Console.Write("Nowa cena: ");
        if (double.TryParse(Console.ReadLine(), out double cena))
        {
            manager.EdytujProdukt(id, nazwa, cena);
            Console.WriteLine("Zmieniono.");
        }
    }
    Console.ReadLine();
}

1 odwołanie
static void UsunProduktInterfejs(SklepManager manager)
{
    Console.WriteLine("\n--- USUWANIE ---");
    Console.Write("Podaj ID: ");
    if (int.TryParse(Console.ReadLine(), out int id)) manager.UsunProdukt(id);
    Console.ReadLine();
}
}
```

.

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

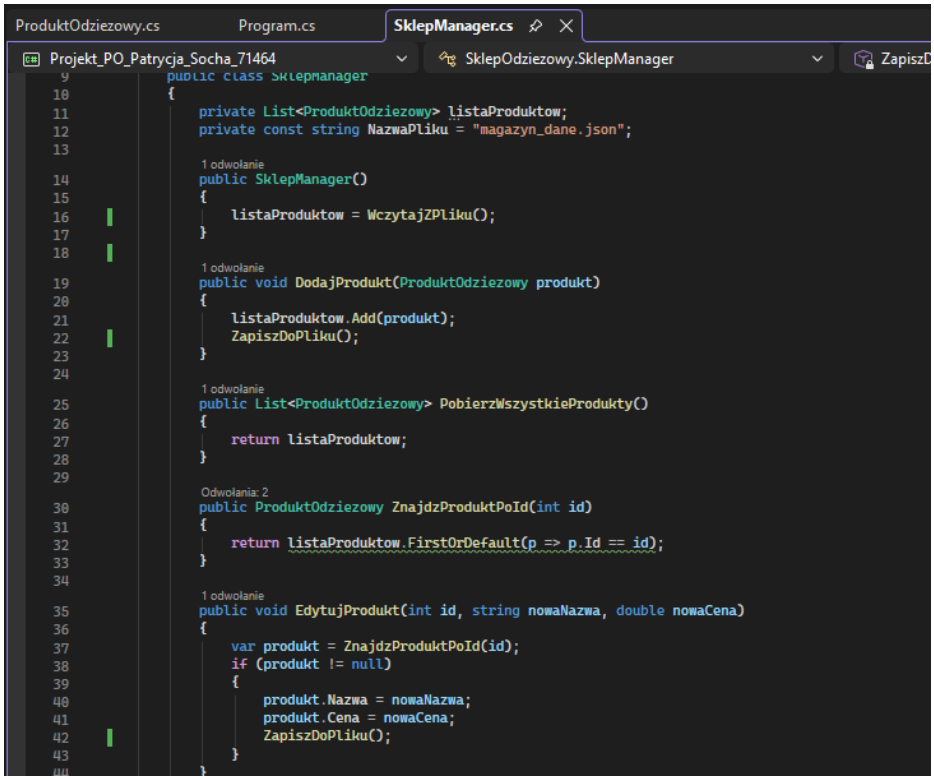
Kluczowym elementem systemu jest klasa `SklepManager`. Przechowuje ona listę obiektów w pamięci RAM, wykorzystując generyczną kolekcję `List<ProduktOdziezowy>`. Dzięki wykorzystaniu biblioteki `System.Linq`, operacje wyszukiwania produktów (np. po ID) są realizowane wydajnie za pomocą metody `FirstOrDefault`.

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.



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