Dokumentacja Techniczna

System "Asystent Zakupowy"

Aplikacja konsolowa oraz WPF w technologii .NET 8.0

Spis treści

1 Wstęp

1.1 Cel Projektu

Celem projektu jest stworzenie narzędzia wspomagającego decyzje zakupowe konsumenta ("Asystent Zakupowy"). System rozwiązuje problem optymalizacji kosztów koszyka zakupowego w obliczu rozproszonych ofert handlowych, różnorodnych gramatur produktów (problem "Unit Pricing") oraz ukrytych kosztów (np. dostawa w sklepach online).

1.2 Zakres Funkcjonalny

System umożliwia:

- Przechowywanie bazy ofert handlowych (Sklep, Produkt, Cena, Promocje).
- Porównywanie cen jednostkowych (zł/kg, zł/l) w celu wykrycia rzeczywistej opłacalności (np. małe opakowanie vs XXL).
- Symulację pełnego koszyka zakupów – algorytm wskazuje sklep, w którym suma paragonu (wliczając dostawę) będzie najniższa.
- Zarządzanie danymi: Dodawanie ofert, Odczyt, Zapis (trwałość danych), Reset do ustawień fabrycznych.

2 Architektura Systemu

2.1 Model Warstwowy (Logiczny)

Aplikacja została zaprojektowana zgodnie z zasadą **Separation of Concerns** (Rozdział Odpowiedzialności). Wyróżniamy trzy logiczne moduły:

1. **Warstwa Prezentacji:** Program.cs (Konsola) oraz MainWindow.xaml (WPF).
2. **Warstwa Logiki Biznesowej:** Offer, Store, ShoppingLogic.
3. **Warstwa Danych:** DataManager, DataSeeder, plik JSON.

2.2 Opis Komponentów

- **Controller / UI (Program.cs):** Odpowiada za interakcję z użytkownikiem, steruje przepływem aplikacji (Load -> Process -> Save) i implementuje pętlę menu.
- **Data Access Layer (DataManager.cs, DataSeeder.cs):**
  - `DataManager`: Odpowiada za serializację i deserializację bazy danych do formatu JSON.
  - `DataSeeder`: Realizuje wzorzec *Factory/Builder* do inicjalizacji bazy danych przy pierwszym uruchomieniu.
- **Domain Model (Core):** Definiuje struktury danych: `Product`, `Offer` oraz abstrakcję `Store`.

3 Zastosowane Koncepcje OOP

Projekt demonstruje praktyczne zastosowanie filarów programowania obiektowego:

3.1 Dziedziczenie i Polimorfizm

Klasa bazowa `Store` definiuje kontrakt dla wszystkich sklepów. Dwa typy sklepów dziedziczą po niej, implementując różną logikę kosztów dodatkowych:

- **Klasa Abstrakcyjna:** `Store`
- **Implementacja 1:** `LocalStore` (Sklep stacjonarny) – nadpisuje metodę kosztów, zwracając 0.00.
- **Implementacja 2:** `OnlineStore` (Sklep internetowy) – nadpisuje metodę kosztów, zwracając stałą opłatę za wysyłkę.

```
// W kodzie klienta wywołanie jest niezależne od typu sklepu:
decimal koszty = sklep.GetAdditionalCost();
// System sam decyduje, czy użyć logiki dla Local czy Online.
```

Listing 1: Przykład polimorfizmu

3.2 Enkapsulacja

Wszystkie klasy domenowe (`Product`, `Offer`) chronią swój stan wewnętrzny poprzez użycie właściwości z prywatnymi setterami (`private set`). Walidacja danych (np. cena > 0) odbywa się w konstruktorach.

4 Algorytmy i Logika Biznesowa

4.1 Algorytm "Smart Unit Pricing"

System rozwiązuje problem nieporównywalnych ofert (np. Cola 0.5L vs 2L) poprzez normalizację ceny do jednostki bazowej.

$$CenaJednostkowa = \frac{CenaOfertowa}{WagaLubObjetosc} \tag{1}$$

Dzięki temu w tabeli okazji system sortuje oferty według parametru `UnitPrice`, a nie ceny przy kasie.

4.2 Algorytm Optymalizacji Koszyka

Funkcja `CalculateBasket` realizuje następującą logikę:

1. **Agregacja:** Grupuje wszystkie dostępne oferty według sklepu.
2. **Matching:** Dla każdego sklepu sprawdza dostępność produktów z listy życzeń.
3. **Selekcja:** Jeśli sklep oferuje kilka wariantów, wybierana jest najtańsza oferta nominalna.
4. **Kalkulacja TCO:** Suma produktów + koszt dostawy.
5. **Ranking:** Sortowanie rosnące według sumy końcowej.

5 Warstwa Infrastruktury i Danych

Ta warstwa odpowiada za komunikację z systemem plików.

5.1 Menedżer Danych (DataManager)

Statyczna klasa narzędziowa realizująca wzorzec **Repository**.

- **Technologia:** `System.Text.Json`.
- **Format:** Plik `oferty.json` z wcięciami (Pretty Print).
- **Fail-Safe:** W przypadku braku pliku zwraca pustą listę, zamiast rzucać wyjątek.

5.2 Inicjalizator Danych (DataSeeder)

Generuje zestaw danych startowych (tzw. seed), zawierający pułapki cenowe (małe opakowania vs XXL) oraz różne promocje w konkurencyjnych sklepach.

6 Warstwa Prezentacji - WPF (GUI)

Oprócz interfejsu konsolowego, system posiada interfejs graficzny w technologii WPF.

6.1 Architektura Widoku

Główne okno `MainWindow.xaml.cs` pełni rolę Code-Behind.

- **Inicjalizacja:** Przy starcie ładuje dane przez `DataManager`.
- **Data Binding:** Metody `UpdateGrid` wiążą listy obiektów z kontrolkami `DataGrid`.

6.2 Funkcjonalności Graficzne

- **Kreator Listy:** Dwukrotne kliknięcie na produkt dodaje go do listy zakupów.
- **Interaktywny Wybór:** Kliknięcie na wynik sklepu podświetla go na niebiesko i aktywuje przycisk płatności.
- **CRUD:** Panel boczny umożliwia dodawanie ofert z walidacją danych wejściowych.

7 Podsumowanie

Projekt "Asystent Zakupowy"jest kompletnym rozwiązaniem programistycznym, demonstrującym czystą architekturę (separacja logiki od UI), zaawansowane OOP (polimorfizm, enkapsulacja) oraz praktyczne algorytmy optymalizacyjne. System jest skalowalny i gotowy do dalszego rozwoju.