PROI 2021L
Projekt - Lista Zakupów

Jakub Sitarek 310892
Jeremi Sobierski 310901
Mateusz Krakowski
Maciej Szczepański

Funkcjonalność programu:

Program udostępnia cyfrowe listy zakupów. Użytkownik komponuje swoje listy zakupów z pomocą interfejsu w terminalu, a program je przechowuje, pozwala na łatwy ponowny dostęp, oraz wylicza sumaryczną wagę i cenę produktów jako dodatkowe informacje, potencjalnie przydatne przed zakupami.

Zaimplementowane klasy:

Product

Klasa reprezentująca interfejs dostępny produkt

1. Specjalizacja **Food** - produkt spożywczy
2. Specjalizacja **Item** - pozostałe produkty
3. **name** - nazwa produktu
4. **weight_gr** - waga produktu w gramach
5. **price_gr** - cena produktu w groszach

List

Klasa reprezentująca listę zakupów

1. **Name -** nazwa listy
2. **Products -** słownik przechowujący produkty

Klasa UI

Klasa interfejsu terminalowego aplikacji

1. **Command** - Wspólny interfejs dla klas komend udostępnianych przez interfejs terminalowy.
2. **Command_help** - komenda wyświetlająca komunikat na temat dostępnych komend
3. **registerCommand** - rejestruje komendę w interfejsie
4. **run** - uruchamia interfejs

Application

Klasa zarządzającej aplikacji

1. **Command_*** - klasy komend do dostępnych w aplikacji, dziedziczą po UI::Command
2. **interface** - referencja na obiekt interfejsu z którego aplikacja korzysta
3. **lists** - zarejestrowane listy zakupów
4. **products** - lista zarejestrowanych produktów
5. **run** - uruchamia aplikację

Komendy są deklarowane oraz rejestrowane za pomocą makra COMMAND(name). Makra te zawarte są w pliku commands.cmds. Definicja komend odbywa się makrem DEFINE_CMD(name, help) { … }. Takie rozwiązanie pozwala łatwo dodawać nowe komendy do interfejsu aplikacji, w razie potrzeby.

FileParser

Klasa obsługująca wymianę danych z plikami

1. **basePath** - obiekt przechowujący ścieżkę katalogu wywołania
2. **app -** referencja na obiekt klasy Application

Dane są uporządkowane w hierarchicznej strukturze folderów. Każdy obiekt przechowywany jest w osobnym pliku, w czytelnej dla człowieka konwencji. Dzięki temu łatwo rozszerzyć oferowaną listę produktów o nowe pozycje i kategorie.

AppException

Klasa bazowa dla wyjątków rzucanych przez funkcjonalności aplikacji

Obsługa:

Utworzenie pliku aplikacji:

make shoppingList

Uruchomienie testów jednostkowych:

Testy powstały z wykorzystaniem frameworku dostarczanego przez Microsoft. Uruchamia się je z poziomu Visual Studio.

Po wpisaniu polecenia “help” wyświetlają się informacje na temat wszystkich dostępnych w aplikacji komend, oraz ich użycia i funkcji. Interfejs pozwala na “zablokowanie” komendy: jeśli podamy komendę która przyjmuje argumenty bez nich, to będziemy mogli skorzystać z niej wielokrotnie, wpisując jedynie argumenty. Pozwala to na przykład na dodawanie wielu produktów bez konieczności powtarzania nazwy komendy “add”.

Pliki list aplikacja generuje sama z siebie, jednak napisana jest tak by były one czytelne także dla użytkownika.

Lista dostępnych produktów jest zamknięta.

Przykładowe wykorzystanie programu:

```
(No list selected) > new MojaLista
List MojaLista created.
MojaLista > add dom
Product named "dom" does not exist
MojaLista > add czapka
czapka added.
MojaLista > add 4 maslo
4x maslo added.
MojaLista > count 5 czapka
Count of czapka set to 5
MojaLista > remove maslo
maslo removed.
MojaLista > print
5 czapka

Total price is .50 zl
Total weight is 1.00 kg

MojaLista > help
lists  - prints all list names

info (product name) - prints info about the product

products  - prints all products

remove (product name) - removed product from selected list

select (list name) - select a list
```

Podział Odpowiedzialności:

Klasy Product, Item, Food - Maciej Szczepański
Klasa List - Mateusz Krakowski
Klasy UI, Application, AppException, Stack - Jeremi Sobierski
Klasa FileParser - Jakub Sitarek