AKADEMIA HANDLOWA
NAUK STOSOWANYCH W RADOMIU

RADOM
ACADEMY OF ECONOMICS

Wydział Studiów Strategicznych i Technicznych

Kierunek: Informatyka, rok II, semestr III (2021/2022)

LABORATORIUM NR 1

ZAAWANSOWANE METODY PROGRAMOWANIA OBIEKTOWEGO

Prowadzący: dr Piotr Dobosz

Zespół laboratoryjny:

Magdalena Szafrańska, nr albumu: 18345

Spis treści

Cel ćwiczenia

Niniejsze laboratorium ma na celu zapoznanie z technikami tworzenia projektów informatycznych. Obecnie projekty tworzone są wedle jednego z wzorców projektowych oraz z wykorzystaniem odpowiednich narzędzi ułatwiających współpracę grupową.

Informacje wstępne

Programowanie obiektowe

Programowanie obiektowe jest coraz powszechniejsze przy tworzeniu projektów. Charakteryzuje się tym, że program jest traktowany jako zbiór współpracujących ze sobą obiektów. Obiekty te łączą stan (czyli zawarte w nich dane) oraz zachowanie (czyli wykonywane na nich operacje). Główne cechy programowania obiektowego to: abstrakcja, hermetyzacja (enkapsulacja), polimorfizm i dziedziczenie.

Na jego popularność wpływa fakt, że niesamowicie usprawnia pracę programistów m.in. przez:

- wielokrotne wykorzystanie tego samego kodu a przez to również zasady DRY (Don't Repeat Yourself)
- niższy koszt rozwoju i konserwacji aplikacji
- stabilność i elastyczność czyli łatwość w dokonywaniu zmian oraz możliwości rozbudowy o nowe funkcje (dzięki dziedziczeniu)
- łatwość przyswojenia ideologii nowego projektu dla wdrażającego się programisty
- ma na celu odzwierciedlać rzeczywistość - ludzki mózg jest bardzo dobrze przystosowany do przetwarzania informacji właśnie w taki sposób.

Do wad programowania obiektowego można za to zaliczyć:

- znaczną ilość nadmiarowego kodu (przy tworzeniu uniwersalnych elementów),
- znaczną komplikację struktury projektu (w przypadku dużych projektów może to być nawet kilkadziesiąt-kilkaset plików),
- możliwą duplikację działań niektórych funkcjonalności (w przypadku dużych projektów taka możliwość znacznie wzrasta).

Wzorce programistyczne i architektoniczne

Obecnie większość aplikacji użytkowych tworzonych jest w językach obiektowych, a przynajmniej w językach, które mają właściwości języków obiektowych.
Pierwszym z etapów na poprawę czytelności projektu jest wykorzystanie tzw. paradygmatów programowania (**wzorce programistyczne**). Dzięki utrzymywaniu kodu w określonej notacji (funkcyjnej, obiektowej, modułowej itp.) odnalezienie się w kodzie i jego ewentualna modyfikacja stanie się łatwiejsza zarówno dla twórcy rozwiązania, jak i osób z nim współpracujących.
Kolejnym etapem jest wykorzystanie **wzorców architektonicznych**. W przypadku Visual Studio i tworzeniu programów obiektowych najlepiej spisuje się wzorzec MVC, aczkolwiek można wykorzystać jego pochodne i/lub inne wzorce projektowe.

Model-View-Controller (MVC)

MVC wprowadza podział w kodzie źródłowym na trzy sekcje (model, view, controller), dzięki czemu kod ten staje się czytelniejszy i łatwiejszy w późniejszym rozwijaniu projektu.

- **Model** - określa jakie operacje będziemy wykonywać po stronie tzw. backendu
- **View** - to, co użytkownik będzie widział, wykonywane tylko po stronie widoku (frontend)
- **Controller** - spina oba powyższe, czyli widok odwołuje się do funkcji w kontrolerze, a kontroler wykonuje funkcję lub sekwencję funkcji z modelu, żeby uzyskać pożądane efekty i zwrócić je do widoku.

Przykładem może być tu moduł produktu w sklepie internetowym gdzie w modelu będziemy przechowywać wszystkie właściwości produktu, w widoku znajdzie się cała otoczka wizualna a kontroler będzie odpowiadał za reakcję na zdarzenia wywoływane np. przez użytkownika.

Środowisko programistyczne

Ciężko dziś powiedzieć, które środowisko jest zintegrowane, a które nie, bo oba mają szereg narzędzi i ta granica jest dość cienka (możemy dzięki Nuget doinstalować dodatki i już ze zwykłego środowiska zrobi się zintegrowane). Osobiście na co dzień używam Visual Studio jako mojego głównego IDE. Najważniejsze cechy środowiska programistycznego to:

- Wsparcie kompilacji i uruchamiania oraz wyboru kompilatora
- Wsparcie dla czytelności kodu
- Posiada elementy ułatwiające pisanie
- Posiada debugger
- Pozwala na instalację dodatkowych zewnętrznych modułów
- Funkcja refactoringu

Repozytorium projektu

Klasycznie nasz projekt tworzony jest w odpowiednim katalogu, który dostępny jest na naszym magazynie danych. Jeżeli chcemy go przenieść/dać komuś kopię jesteśmy zmuszeni kopiować cały katalog projektu, a co najmniej pliki z kodem źródłowym. Ponadto przy dużych projektach zmiany dokonane w kodzie mogą dopiero po jakimś czasie wykazać nieprawidłowości w funkcjonowaniu – wszystko zależy od czasu dokonywania testów aplikacji.
Obecnie najlepszym wyborem jest GIT. Pozwala on na śledzenie niemal każdej zmiany w edytowanych plikach – zapisuje każdą nową wersję pliku jako nowy, osobny plik lub jeżeli mamy do czynienia z plikiem kodu źródłowego to zapis następuje tylko dla zmienianych linii kodu (nadpis różnicowy). Dzięki temu, jeżeli wcześniejsza wersja kodu spełniała nasze oczekiwania (a aktualna zawiera błędy) bez problemu będziemy mogli cofnąć się do właściwej wersji kodu by ponownie rozpocząć modyfikację (lub będziemy w stanie skuteczniej ustalić źródło ewentualnego błędu).

Użyte narzędzia i technologie

- Windows 10 Home (wersja 21H1)
- Microsoft Visual Studio Community 2019 (wersja 16.11.1)
- Repozytorium kodu GIT na Github (link do repozytorium: znajduje się tutaj)
- Język programowania C#
- Platforma .NET 5.0.

Przebieg laboratorium

Laboratorium polega na utworzeniu dowolnego projektu oprogramowania z wykorzystaniem wszystkich opisanych wyżej rozwiązań i narzędzi. W moim projekcie stworzyłam program, który pozwala tworzyć tzw. listy TODO, czyli tworzy i zarządza listą rzeczy do zrobienia.
Do głównych funkcjonalności programu należy zaliczyć możliwość przechowywania danych w pliku, dostępnych również po zamknięciu aplikacji. Dane te zapisywane są w sposób dynamiczny, bez ingerencji użytkownika. Po ponownym uruchomieniu aplikacji dane są dopisywane do istniejącego pliku - o ile takowy już się znajduje.
Dodatkowo, program jest w pełni funkcjonalny - nie następuje jego zamknięcie po wykonaniu pojedynczej akcji. Stale, na każdym etapie, mamy dostęp do wszystkich funkcji bez konieczności ponownego uruchomienia aplikacji.

Kod źródłowy

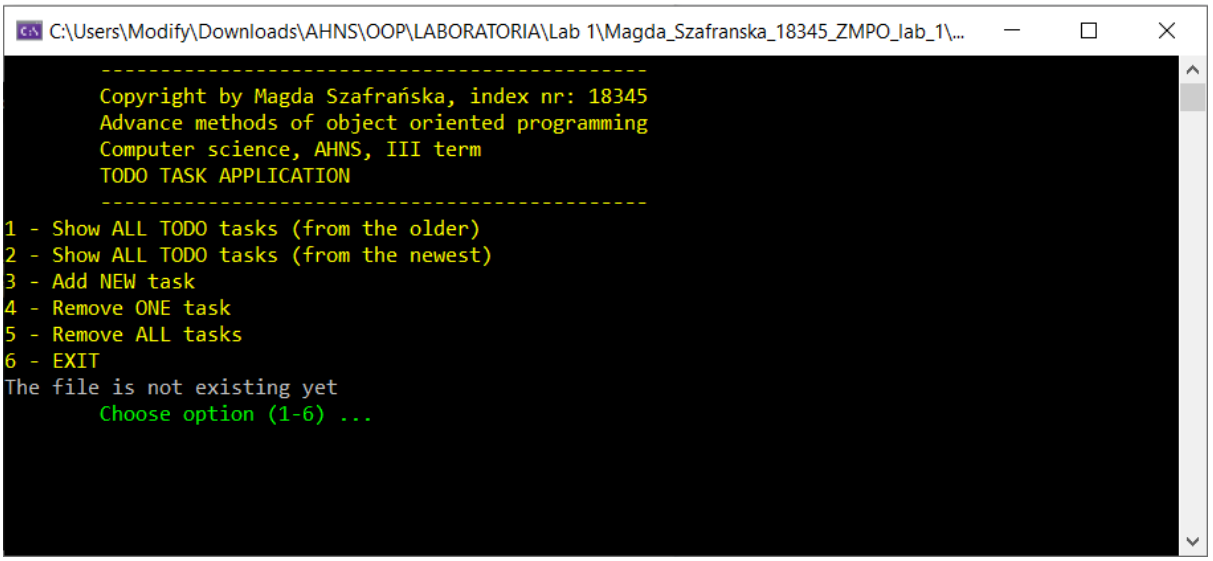
Kod źródłowy całego projektu umieściłam w repozytorium zdalnym na GitHubie. Link do niego: znajduje się tutaj.
Ze względu na liczne klasy zawierające się w osobnych plikach, zdalna forma wydaje się najodpowiedniejsza w tym przypadku. W dalszej części sprawozdania zamieszczę zrzuty ekranu jedynie z funkcjonalności aplikacji w działaniu.

Założenia aplikacji

Aplikacja TODO zawiera następujące funkcje:

1. Wyświetlanie wszystkich zadań w porządku od najstarszych
2. Wyświetlanie wszystkich zadań w porządku od ostatnio dodanych
3. Dodanie nowego zadania
4. Usunięcie jednego wybranego zadania
5. Usunięcie wszystkich zadań z listy
6. Zakończenie działania programu

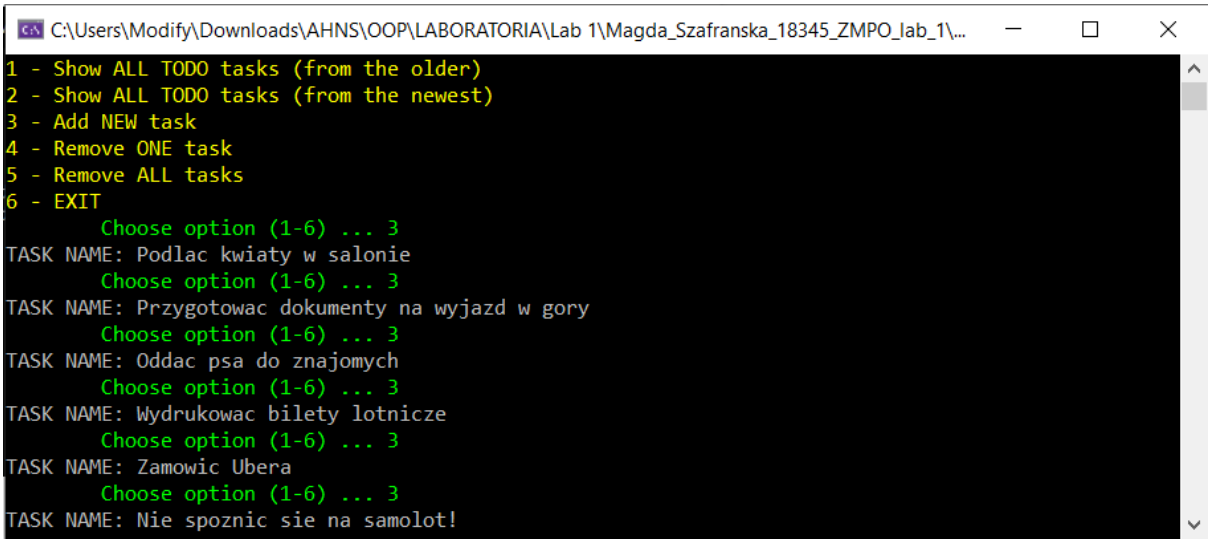
Wszystkie założenia zawarłam w menu, które ukazuje się na początku po uruchomieniu programu. Poniżej screen zaraz po uruchomieniu pliku wykonywalnego.



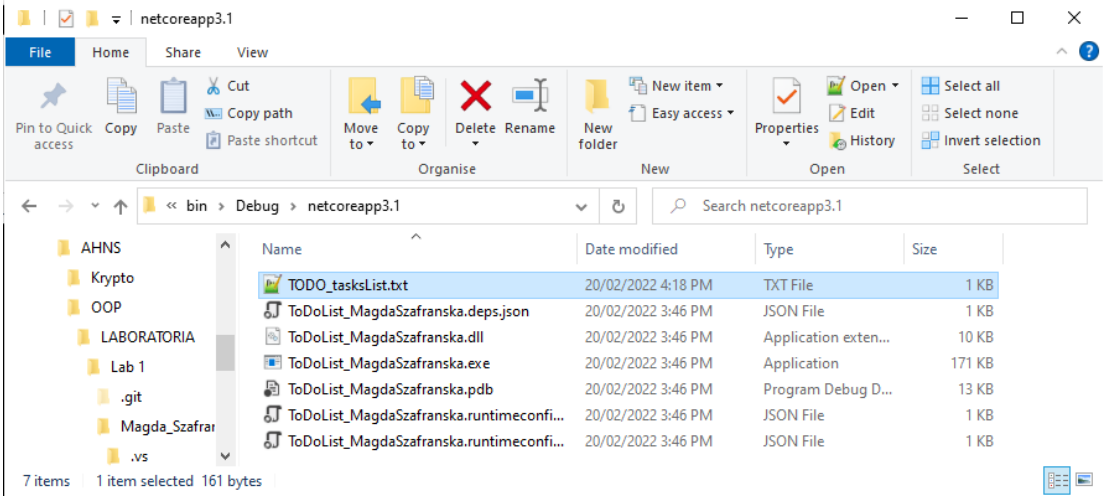
Funkcjonalność

1. Dodanie nowego zadania

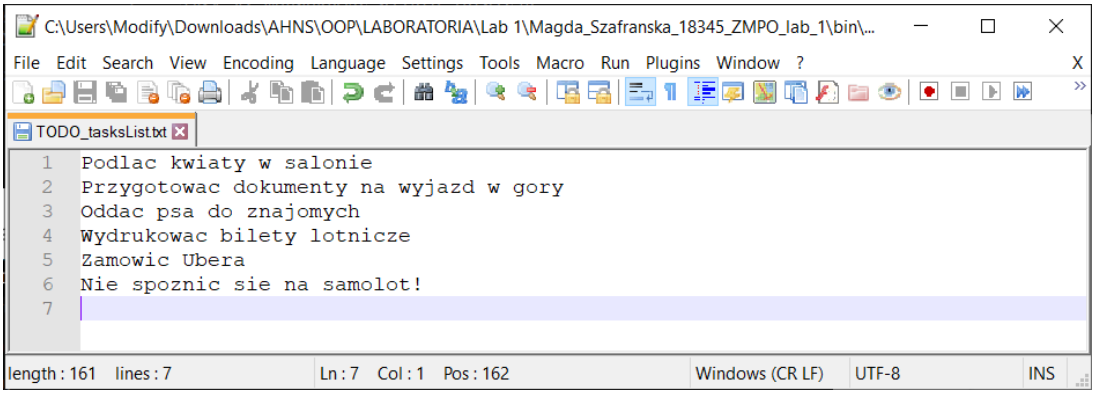
Do zaprezentowania funkcjonalności dodałam 6 zadań wybierając z menu opcję numer 3.



W folderze z plikiem wykonywalnym zostaje automatycznie utworzony plik tekstowy. Jeśli takowy już istnieje (z poprzedniego uruchomienia programu), dane zostają dopisywane. Poniżej zrzut z utworzonym plikiem.

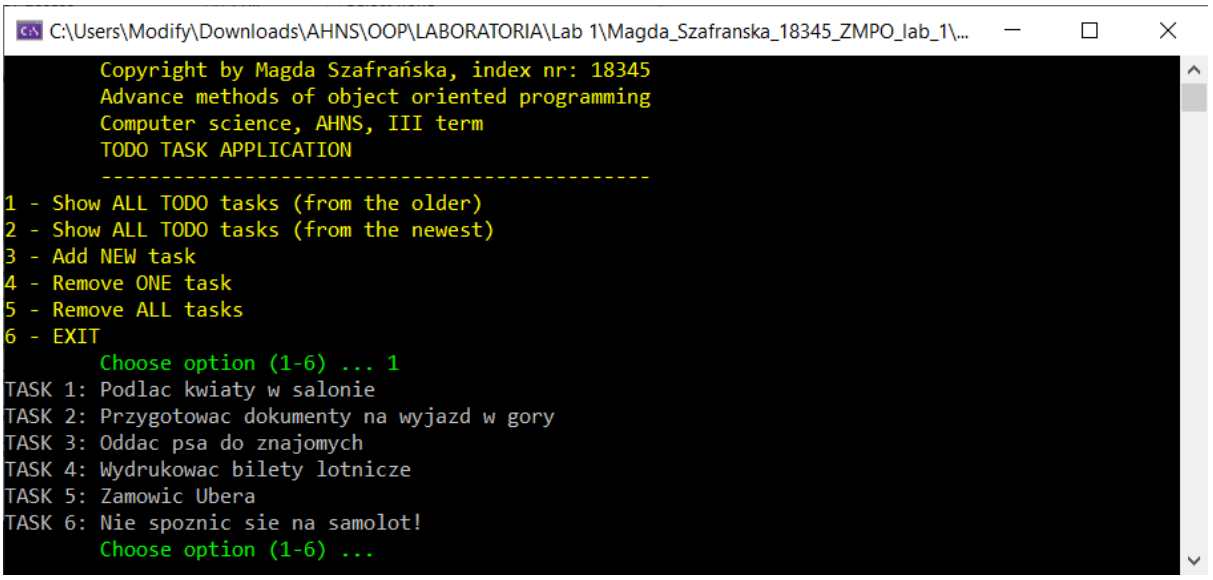


Dodatkowo plik tekstowy z zapisanymi świeżo dodanymi zadaniami.



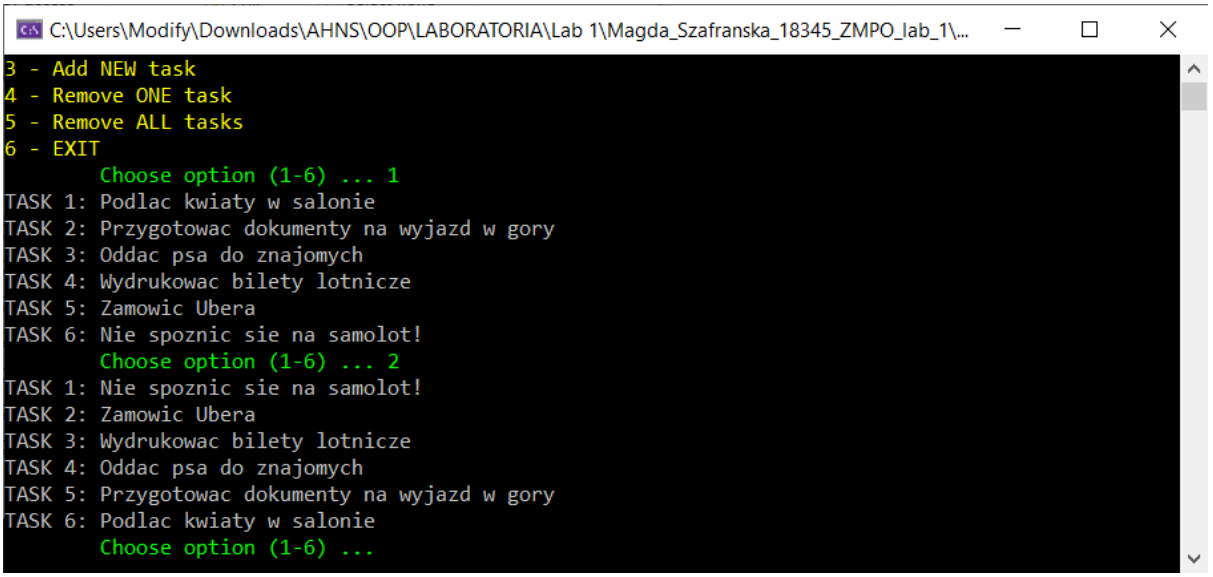
2. Wyświetlanie wszystkich zadań (od najstarszych)

Sortowanie dodanych zadań według kolejności od najstarszych do ostatnio dodanych.



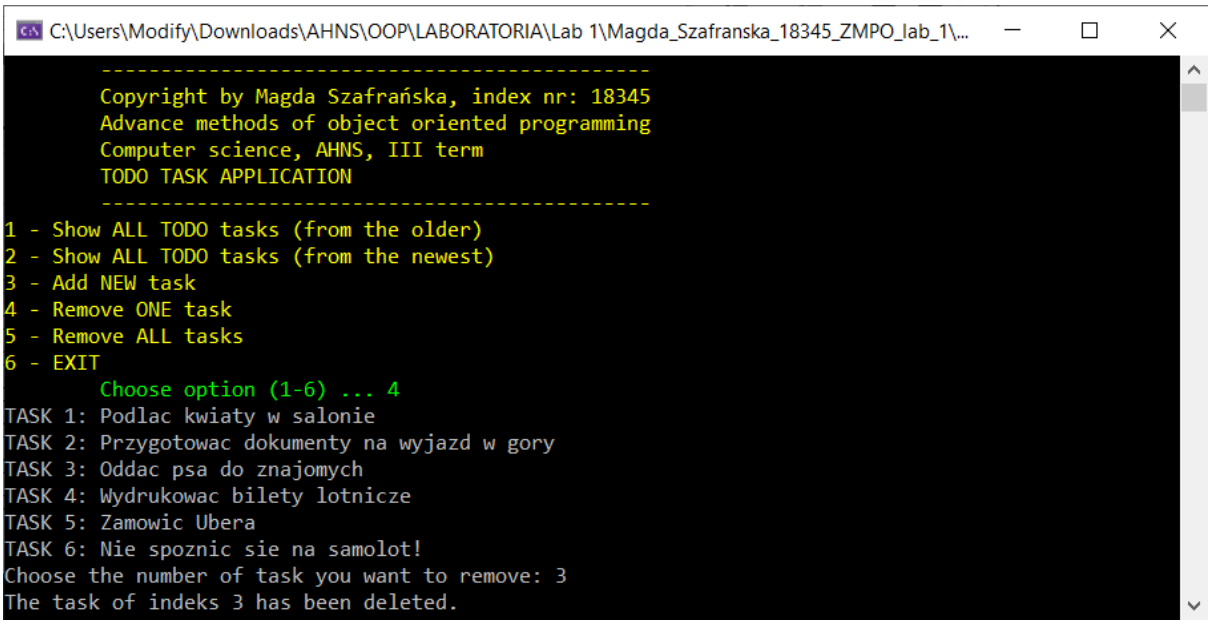
3. Wyświetlanie wszystkich zadań (od ostatnio dodanych)

Sortowanie dodanych zadań według kolejności od ostatnio dodanych do najstarszych. Na poniższym zrzucie ekranu widać dodatkowo różnicę pomiędzy tym sortowaniem oraz poprzednim (od najstarszych).

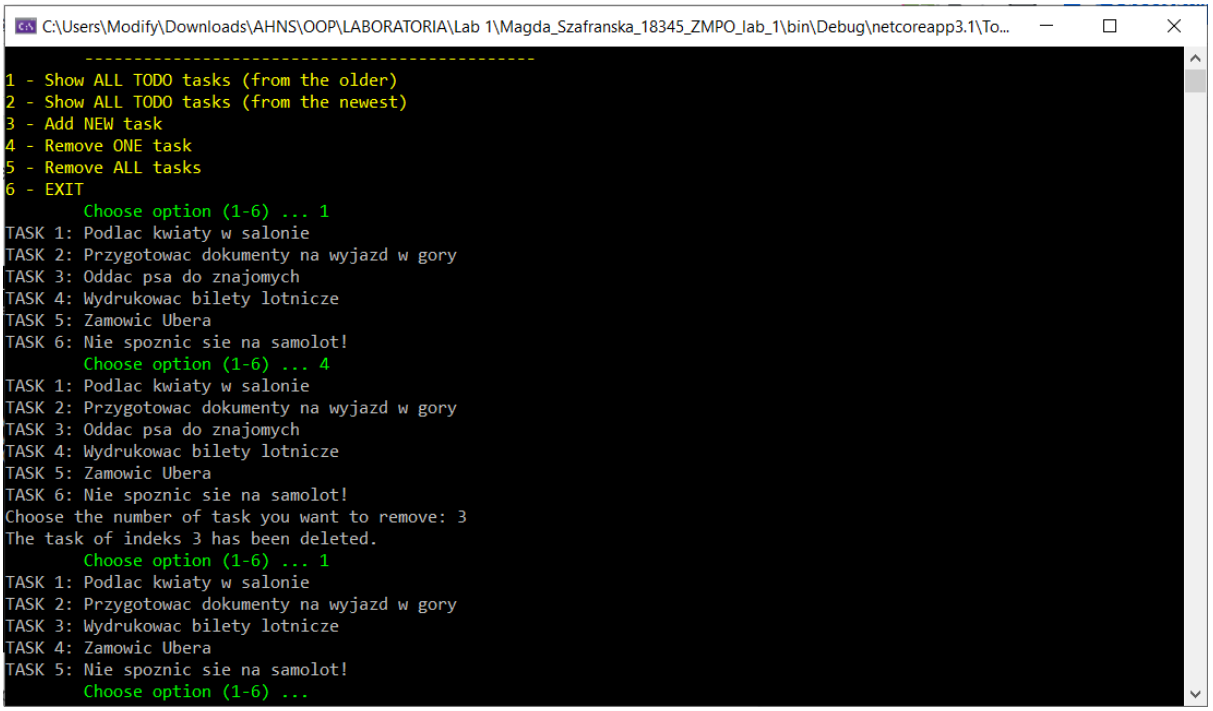


4. Usunięcie jednego wybranego zadania

Korzystając z opcji numer 4 w menu, widzę wszystkie obecne zadania. Program pyta, które z nich chcę usunąć. Usuwam zadanie o numerze 3.

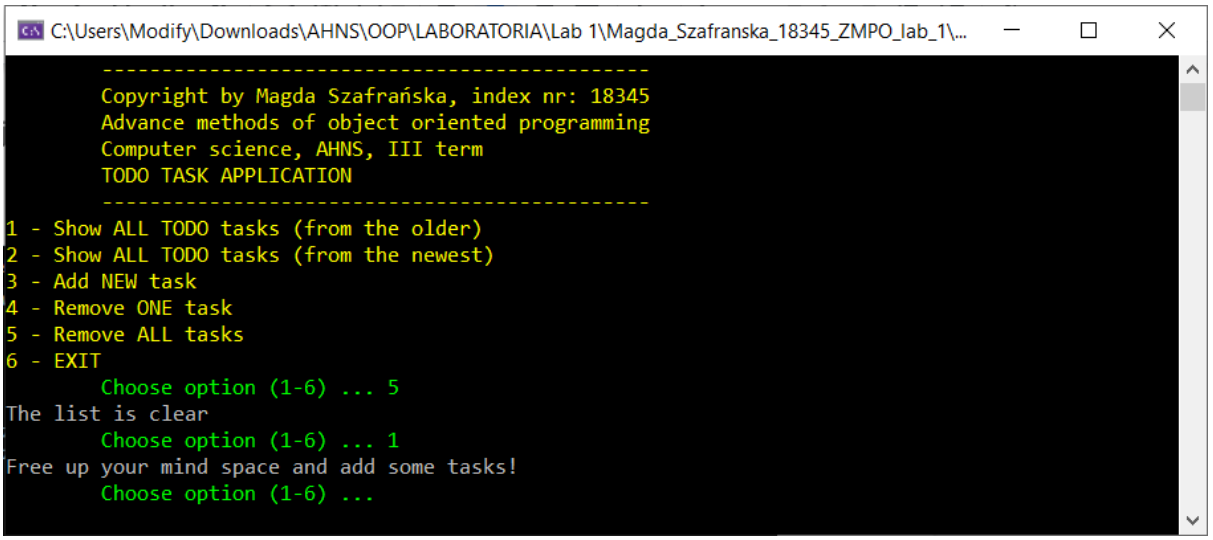


Po tej operacji wyświetlam wszystkie taski aby sprawdzić, czy wskazane zadanie zostało usunięte. Faktycznie, wybrane zadanie (o numerze 3) zostało usunięte z listy.

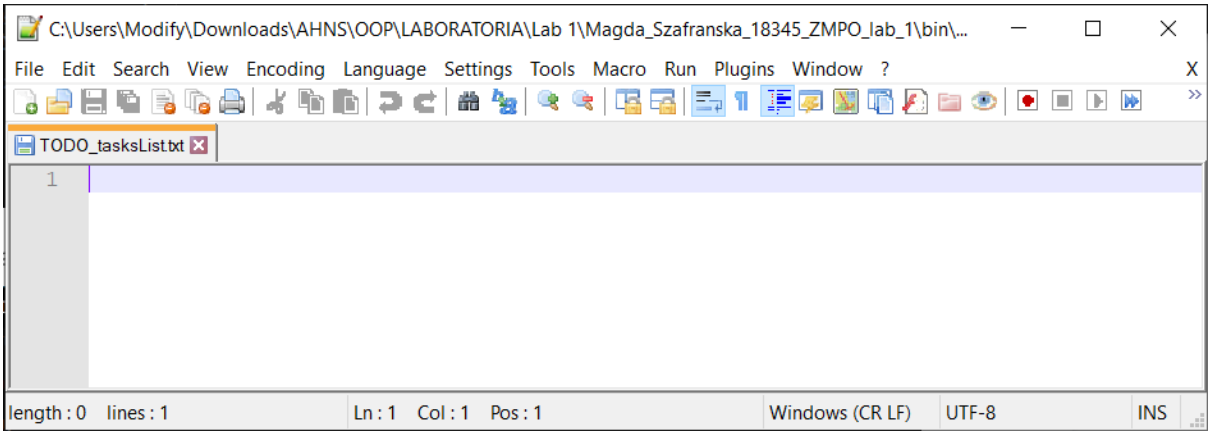


5. Usunięcie wszystkich zadań z listy

Opcja numer 5 w menu usuwa wszystkie utworzone i zapisywane na bieżąco do pliku zadania. Widzimy również stosowny komunikat o tym, że lista z zadaniami jest pusta.

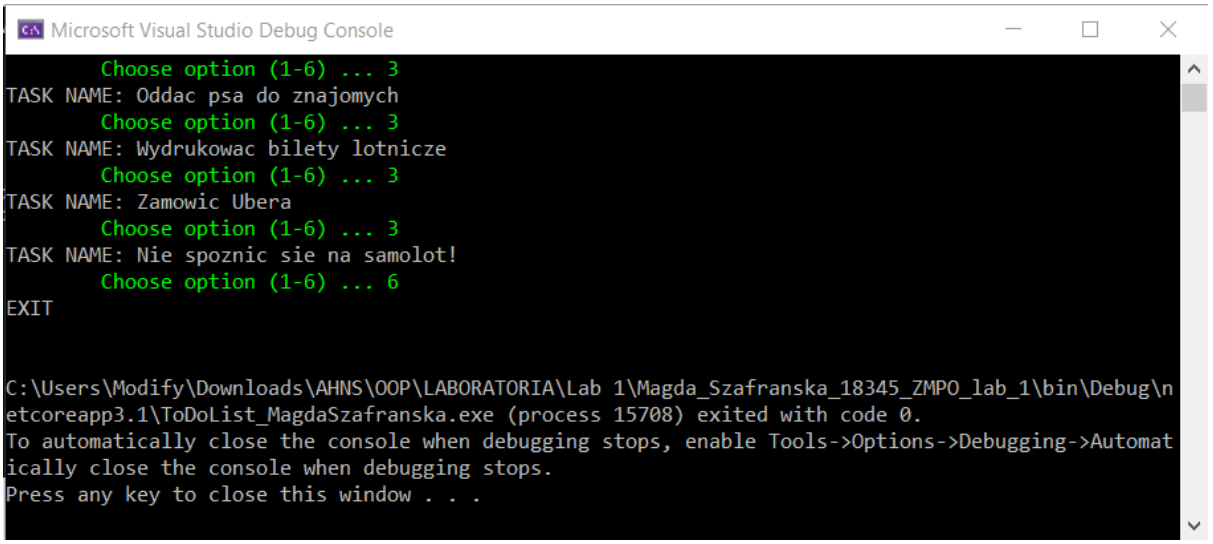


Operacja ta spowodowała także oczekiwaną zmianę w pliku tekstowym, mianowicie usunięcie z niego wszystkich zadań. Sam plik pozostaje w folderze więc przy ponownych operacjach na liście nie będzie na nowo tworzony a jedynie dopisywane będą do niego kolejne zadania.



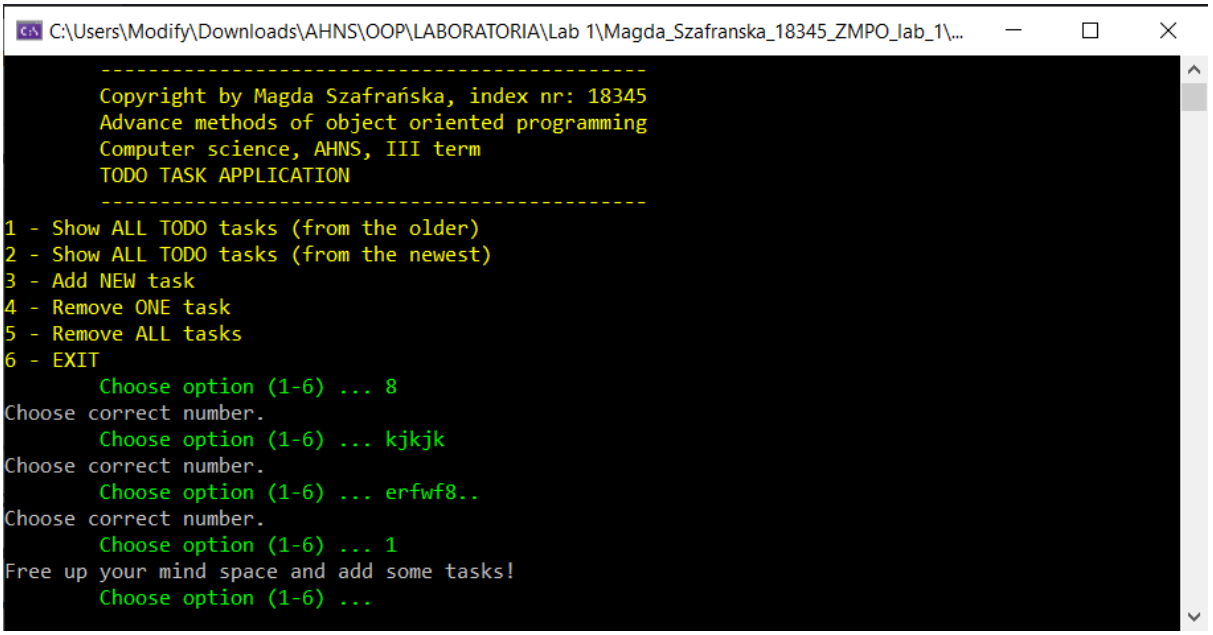
6. Zakończenie działania programu

Wybranie numeru 6 z menu powoduje wyświetlenie w konsoli komunikatu i wyjście z programu. Zamknięcie okna konsoli wymaga naciśnięcie dowolnego klawisza.



7. Obsługa niewłaściwego wprowadzania znaków

Program jest odporny na wprowadzenie innych niż oczekiwane znaków z klawiatury. Dowolna kombinacja powoduje wyświetlenie stosownego komunikatu i daje możliwość ponownego wprowadzenia znaku.


C:\Users\Modify\Downloads\AHNS\OOP\LABORATORIA\Lab 1\Magda_Szafranska_18345_ZMPO_lab_1\...
Copyright by Magda Szafrańska, index nr: 18345
Advance methods of object oriented programming
Computer science, AHNS, III term
TODO TASK APPLICATION
1 - Show ALL TODO tasks (from the older)
2 - Show ALL TODO tasks (from the newest)
3 - Add NEW task
4 - Remove ONE task
5 - Remove ALL tasks
6 - EXIT
Choose option (1-6) ... 8
Choose correct number.
Choose option (1-6) ... kjkjk
Choose correct number.
Choose option (1-6) ... erfwf8..
Choose correct number.
Choose option (1-6) ... 1
Free up your mind space and add some tasks!
Choose option (1-6) ...

Wnioski

Zastosowanie programowania obiektowego w projekcie udowodniło niesamowitą łatwość w potencjalnej dalszej rozbudowie. Wykorzystanie jego założeń bardzo ułatwi mi dalszą rozbudowę projektu o nowe funkcjonalności, jeśli zajdzie taka potrzeba w przyszłości.

Udało mi się zawrzeć wszystkie technologie i założenia planowanej funkcjonalności aplikacji.Podział na klasy idealnie odwzorowuje opisywaną we wstępie przydatność szczególnie w pracy zespołowej, kiedy członkowie zespołu mogą pracować równocześnie, niezależnie od siebie.