Treasury

Symulator urzędu skarbowego

Łukasz Szarejko Bartosz Latosek Marcin Grabysz

Semestr letni 2021

Spis treści

1 Skład zespołu i zakres odpowiedzialności

- Szarejko Łukasz - odpowiedzialny za klasy rządzące programem: Treasury, Calculator, ConsoleInterface oraz MainApp
- Latosek Bartosz - odpowiedzialny za podstawowe klasy programu takie jak Person oraz obsługę plików *json* (odczyt i zapis danych w klasie Treasury i ConsoleInterface).
- Grabysz Marcin - odpowiedzialny w głównej mierze za napisanie systemu rozliczeniowego (klasa Payment oraz jej pochodne) oraz projektowanie rozwiązania.

Ponadto cały zespół pracował przy testach jednostkowych i pisaniu dokumentacji.

2 Opis zadania

Należy stworzyć system dla urzędu skarbowego do rozliczania ludzi oraz przedsiębiorstw. Ma wspierać łatwe dodawanie nowych form podatku na dowolny aspekt działalności gospodarczej.

Zadanie celowo nie zostało do końca sprecyzowane. Częścią ćwiczenia jest zaproponowanie własnej interpretacji tematu i skonsultowanie jej z prowadzącym.

1. Podział na podproblemy
2. Kontrola dostępu (podział na elementy publiczne/prywatne)
3. Testy jednostkowe wszystkich istotnych części projektu
4. Dynamiczny polimorfizm
5. Szablony
6. Informacja o podziale odpowiedzialności wewnątrz zespołu
7. Praca ma zostać omówiona z prowadzącym

Zgodnie z naszą interpretacją zadania, program przyjmuje dane o osobach (imię, nazwisko, wiek) oraz o umowach które zawarły (typ umowy, kwota). Dane są czytane z zewnętrznego pliku po uruchomieniu programu, lecz mogą być także wprowadzane i edytowane przez użytkownika za pomocą prostego interfejsu konsolowego.

Po załadowaniu wszystkich danych, program może rozpocząć kalkulacje. Na podstawie kwoty, typu umowy, wieku osoby czy już osiągniętych przychodów obliczona zostaje wysokość następujących płatności pobieranych przez Urząd Skarbowy i Zakład Ubezpieczeń Społecznych: składka emerytalna, składka rentowa, składka chorobowa, składka zdrowotna oraz podatek dochodowy.

Otrzymane dane zostają zapisane w plikach.

3 Instrukcja obsługi programu

3.1 Wersja DEBUG

Do uruchomienia programu w tej wersji należy mieć zainstalowane środowisko typu Visual Studio 2019. Należy ustawić MainApp jako StartUp project i uruchomić kompilator. Na tym etapie powinna pojawić się konsola z programem. Do zaimportowania danych potrzebujemy przygotowanego pliku *json* z danymi osób (my oferujemy wstępny plik `people.json`), nazwę takiego pliku należy podać po uruchomieniu programu.

3.2 Wersja RELEASE

Aby uruchomić program w wersji release, należy po pobraniu repozytorium wejść do folderu `x64/Release` i odpalić tam `MainApp.exe`. Spowoduje to włączenie konsoli z programem. W tym folderze znajdują się pliki z danymi do zaimportowania (`people.json`) oraz będą się tu pojawiać wygenerowane pliki *json*.

3.3 Dalsza obsługa

Po zaimportowaniu wstępnych danych na konsoli powinny ukazać się możliwe opcje do wyboru, są to kolejno:

1. Wygenerowanie całego sprawozdania podatkowego dla wszystkich osób które znajdują się aktualnie we wpisie urzędowym. Wygenerowany plik nosi nazwę `List_Settlement.json` i znajduje się w tym samym folderze co aplikacja.

2. Wygenerowanie sprawozdania podatkowego dla konkretnej osoby z listy, która ukaże się po wybraniu tej opcji. Wygenerowany plik będzie nosił nazwę `nazwisko_imie.json`.

3. Dodanie osoby do wpisu urzędowego. Program poprosi użytkownika o wpisanie imienia, nazwiska oraz wieku osoby. Do dalszej edycji osoby należy wybrać opcję 5.

4. Usuwanie osoby z wpisu urzędowego poprzez wybranie jej z listy, która ukaże się po wybraniu tej opcji.

5. Edytowanie osoby:

   - Usuwanie n-tej umowy z listy danej osoby
   - Dodawanie nowej umowy. Tutaj program poprosi użytkownika o wybranie rodzaju umowy oraz rocznego przychodu w groszach

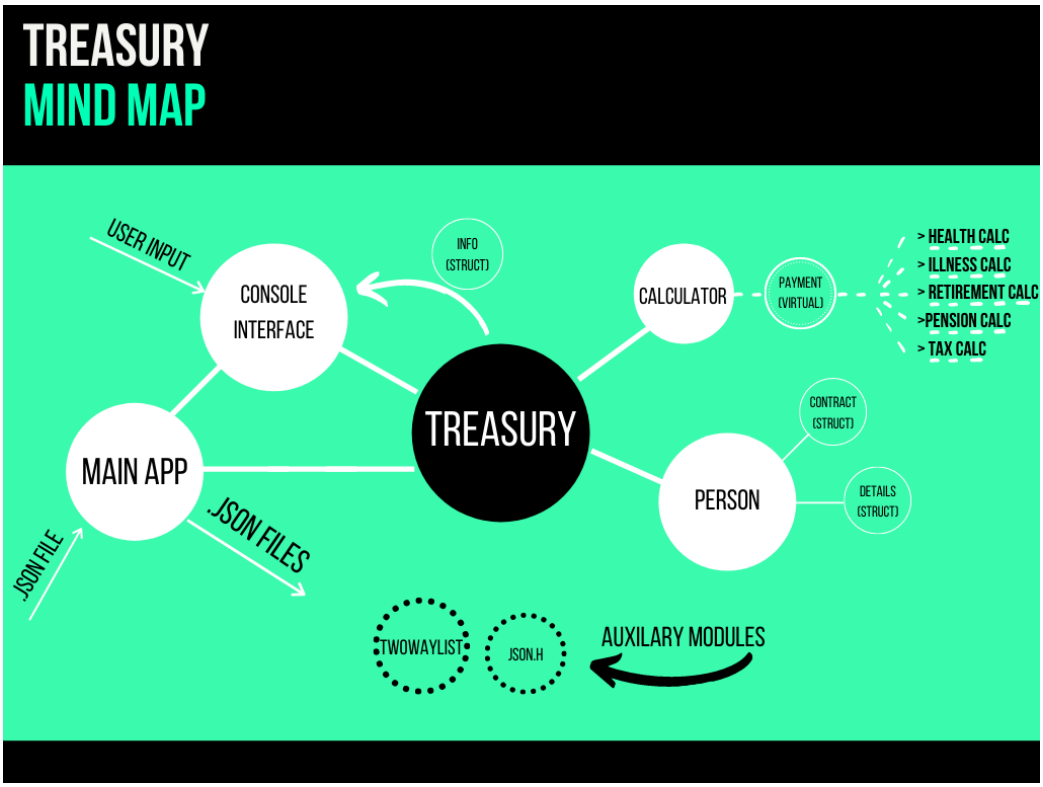
6. Zamknięcie programu. Uwaga! Wszelkie dodane osoby nie znajdą się w pliku importowanym na początku!

Przy błędnym wyborze, użytkownik zostanie poproszony o ponowne dokonanie wyboru.

4 Środowisko aplikacji

Projekt został zrealizowany przy pomocy Visual Studio 2019 oraz GitLaba Politechniki Warszawskiej: Urząd Skarbowy · GitLab (pw.edu.pl)

5 Architektura rozwiązania



Projekt MainApp, jak sama nazwa wskazuje, jest głównym kodem programu, który uruchamia ConsoleInterface i powołuje do życia obiekt klasy Treasury. Gdyby zaszła potrzeba, aby zaimplementować inne rodzaje komunikacji z użytkownikiem, wystarczyłoby tylko ją modyfikować.

Klasa ConsoleInterface jest jedną z możliwych form komunikacji z użytkownikiem. Pozwala na korzystanie z wszelkich funkcji klasy Treasury przez konsolę.

Klasa Treasury przechowuje listę wskaźników na osoby, które są aktualnie w spisie urzędu oraz posiada obiekt klasy Calculator do przeprowadzania obliczeń podatkowo-składkowych. Posiada funkcję pozwalające na zaimportowanie danych z pliku *json*, dodawanie, usuwanie, edytowanie osoby oraz generujące struktury Info, w których są zawarte informacje podatkowe poszczególnych osób, które chcemy udostępniać interfejsom. Docelowo, klasa Treasury pozwala na swobodne tworzenie nowych interfejsów.

Klasa Person reprezentuje osobę podatnika. Posiada następujące atrybuty: imię, nazwisko, wiek, płatności do zapłacenia (początkowe równe 0), zarobki już rozliczone (początkowo równe 0; pewnych składek nie trzeba naliczać, jeżeli płatnik osiągnął już pewien próg zarobków i odprowadził od nich składki), strukturę Details (przechowującą dokładną wartość każdej płatności) oraz listę umów (każda umowa jest strukturą Contract). Person posiada także zestaw potrzebnych getterów i setterów oraz metody pozwalające na dodawanie i usuwanie umów.

Klasa Calculator przechowuje wskazania na obiekty typu Payment oraz podaje naszym kalkulatorom każdą umowę podanej osoby. Klasa została napisana tak, aby można było łatwo dodać kolejne typy podatków lub składek. Nie uwzględnia jednak usuwania ich, ponieważ konsekwentnie trzymaliśmy się stanu faktycznego naszego państwa.

Klasy Retirement, Pension, Illness, Health oraz Tax są pochodnymi wirtualnej klasy Payment. Każda z tych klas posiada jedną metodę - calculate(). Działanie calculate() jest unikalne dla każdej płatności - każda składka czy podatek przewiduje inny procent oraz inne ulgi (ze względu na wiek płatnika, roczne zarobki czy typ umowy).

Chociaż program nie przewiduje dodawania nowych form płatności przez użytkownika, taką aktualizację można przeprowadzić bardzo szybko, dzięki zastosowaniu dynamicznego polimorfizmu. Jeżeli zajdzie potrzeba wprowadzenia nowej składki, wystarczy zaimplementować nową klasę pochodną od Payment z odpowiednią klasą calculate() i dodać ją do listy klas obsługiwanych przez klasę Calculator zarządzającą obliczaniem wszystkich płatności.

6 Sposób testowania

Do przetestowania naszego programu, napisaliśmy testy jednostkowe to wszystkich znaczących części projektu. Pominięte zostało napisanie testów jednostkowych dla interfejsu, który łatwiej jest testować podczas korzystania z programu.