Cel projektu

Celem projektu jest napisanie programu do pomocy kasjerom pływalni. Program ma pomagać kasjerom przy wyborze odpowiedniego terminu rezerwacji biletu. Jeśli termin wybrany przez klienta jest niewłaściwy tzn. pływalnia jest w wyznaczonym terminie nieczynna lub pełna ma zaproponować następny możliwy termin. Oprócz pomocy przy terminarzu program ma również sporządzać dniowy raport finansowy na podstawie cennika. Cena biletu jest zależna od typu klienta, godziny i dnia tygodnia.

Moduły i klasy

Program dzieli się na trzy główne moduły importowane do main.py będący modułem pokazujący przykładowe działanie programu, są to:

1. *accountant.py* – jest to moduł odpowiedzialny za sporządzanie raportów finansowych i wyznaczaniu cen biletów. Na końcu modułu znajduje się mały skrypt do testowania poprawności operacji na plikach. W jego skład wchodzą klasy:
   1.1. *PriceList* - zajmuje się interpretacją danych zawartych w cenniku. Na potrzeby projektu jest to głównie klasa pomocnicza dla klasy *Accountant* i nie będzie raczej wykorzystywana przez użytkownika.
   1.2. *Accountant* – jest to klasa stworzona specjalnie do sporządzania raportów finansowych i zapisu ich do pliku. Klasa przy rejestracji transakcji pobiera dane dla określonego miesiąca, uzupełnia go o ilość i przychód ze sprzedaży biletów. Klasa zakłada że podany termin jest możliwy do rezerwacji. Klasa operuje na plikach JSON w trakcie pracy programu dzięki czemu nawet po wyłączeniu skryptu klasa nadal będzie miała zapisane poprzednie transakcje.
2. *scheduler.py* – jest to moduł odpowiedzialny za rezerwowanie terminów biletów w terminarzu i w przypadku niedostępności podanego terminu proponowanie następnego dostępnego terminu. Na końcu modułu znajduje się mały skrypt do testowania poprawności operacji na plikach. W jej skład wchodzą:
   2.1. OpeningHours – jest to klasa pomocnicza dla klasy *Scheduler.* Klasa zajmuje się interpretacją harmonogramu pracy pływalni.
   2.2. Scheduler – klasa zajmująca się organizacją terminarza. Sprawdza czy podany typ biletu jest możliwy do zarezerwowania na podaną godzinę. Jeśli jest to niemożliwe pozwala na wyszukanie następnego terminu kiedy jest to możliwe. Klasa operuje na plikach JSON w trakcie pracy programu dzięki czemu będzie pamiętała o rezerwacjach z np. poprzedniego dnia.
3. *ui.py* – moduł zawierający klasę pomocniczą przy tworzeniu konsolowego UI. W jego skład wchodzi klasa:
   3.1. UI – klasa z funkcjami pomocniczymi dla modułu *main.py*. Minimalizuje ilość kodu potrzebnego w pliku głównym.

Jak używać?

By wprowadzić dane na temat pływalni trzeba stworzyć plik JSON o specjalnej strukturze. Przykładowy plik znajduje się w *data/info.json*, a następnie wpisać ścieżkę do tego pliku jako argument klas Scheduler i *Accountant* w pliku main.py (domyślnie program używa info.json). Następnie uruchomić plik main.py i postępować zgodnie z instrukcjami wyświetlanymi w konsoli.

W skład pliku JSON z danymi o pływalni wchodzą:

1. *NAME* - nazwa pływalni, nie ma zastosowania w kodzie poza wyświetlaniem w UI.

2. *NUMBER_OF_LANES* – liczba torów pływalni. Na jej podstawie wyliczana jest liczba dostępnych biletów
3. *OpeningHours* – słownik gdzie kluczami są dni tygodnia pisane dużą literą po angielsku z listą dwuelementową jako wartość. Pierwszy element listy oznacza godzinę otwarcia, a drugi zamknięcia przy czym zakładane jest że jeśli pływalnia działa od np. 8:00-20:00 to o godzinie 20:00 nie ma wejścia na pływalnie.
4. *PriceList* – słownik słowników będący reprezentacją cennika. W skład cennika wchodzą:
   4.1. *client_age* – słownik podstawowych wartości biletów dla podanych kategorii wiekowych. Program wyróżnia cztery kategorie: dziecko, student, normalny, senior
   4.2. *lane_price* – podstawowa cena wynajmu pojedynczego toru
   4.3. *day_of_entry* – słownik z kluczami takimi jak *OpeningHours*, ale jako wartości posiada współczynnik przez który będzie przemnażana wartość biletu
   4.4. *time_of_entry* – lista słowników gdzie każdy element ma dwa klucze: *hours* będąca listą dwuelementową i *factor* będące liczbą rzeczywistą. Element *hours* podobnie jak w *OpeningHours* oznacza przedział czasowy. W podanym przedziale czasowym obowiązuje podany *factor*, będący współczynnikiem cenowym przez który będzie przemnażana cena biletu.

Refleksje

Pisanie projektu przez większość czasu przebiegało sprawnie chociaż zdarzyło się parę fragmentów kodu na które straciłem dużo więcej czasu niż powinienem.

Z powodu doświadczeń z projektami pisanymi w językach kompilowanych początkowa struktura mojego projektu z perspektywy Pythona była nieprzemyślana. Wciągu pisania kodu wielokrotnie zdarzały mi się błędy związane z importowaniem modułów lub wynikające z drobnych różnic między pracą z interpreterem a kompilatorem. Ostatecznie doprowadziło to do przepisania przeze mnie sporej części projektu.

Drugim elementem który przez długi czas nie dawał mi spokoju są testy jednostkowe. Na relatywnie wczesnym etapie pracy nad projektem postanowiłem oprzeć najważniejsze metody o odczyt i zapis do pliku. Z tego powodu postanowiłem napisać małe skrypty na końcu modułów by przetestować poprawność rozwiązań. Kolejne problemy z testami jednostkowymi wynikały z mojej niewiedzy o PyTest i błędy typu Discovery Error. Z powodu tych błędów ostatecznie połączyłem kilka modułów w jeden ponieważ PyTest miał problemy z relatywną ścieżką do modułów przy imporcie.

Ostatecznie udało mi się zaimplementować funkcje które planowałem i były wymagane na potrzeby projektu. Funkcje programu operują na odczycie i zapisie do pliku dzięki czemu program może być wyłączony nawet w niekonwencjonalny sposób bez ryzyka utraty zarejestrowanych danych. Program tworzy pliki na każdy miesiąc i folder na każdy rok dzięki czemu (w teorii) możliwe jest odczytywanie i zapis raportów oraz terminarzy z nieskończonej liczby dni (UI tego nie obsługuje ale metody klas jak najbardziej)