Dokumentacja projektu PIPR

Treść zadania:

Napisz program do obsługi pływalni. Pływalnia ma nazwę, godziny pracy, cennik oraz określoną liczbę torów. Są dwa rodzaje klientów na pływalni: szkółki pływackie i klienci indywidualni. Klienci mogą rezerwować bilety na pływalnie na wyznaczoną godzinę. Minimalny czas rezerwacji to jedna godzina. Maksymalna liczba dostępnych biletów jest równa pięć razy liczba wolnych torów. Szkółki pływackie mogą rezerwować cały tor. W jednym momencie szkółki pływackie nie mogą zarezerwować więcej niż 35% wszystkich torów.

Program obsługujący pływalnie powinien umożliwiać rezerwację biletów dla klientów indywidualnych oraz rezerwację torów dla szkółek pływackich z uwzględnieniem dostępności i warunków opisanych powyżej. W przypadku braku miejsc na wybrany przez klienta termin, program powinien zaproponować najbliższy możliwy termin. Program ma przechowywać historię rezerwacji oraz na koniec dnia tworzyć raport finansowy, pokazujący jaki jest przychód pływalni danego dnia. Cennik pływalni powinien być zróżnicowany, w zależności od pory dnia, dnia tygodnia oraz rodzaju klienta.

Spis treści:

Kod źródłowy rozwiązania będzie realizowany w języku Python.

Wykorzystane biblioteki:

W programie zostały wykorzystane następujące biblioteki:

- Calendar - najważniejsza biblioteka w kontekście tworzenia historii rezerwacji oraz sprawdzania ceny rezerwacji, mianowicie zawarte w niej funkcje pomogły stworzyć plik json, który imituje bazę danych oraz zawiera rzeczywiste daty – tzn zawiera dokładną długość dni w poszczególnych miesiącach. Przy sprawdzaniu ceny biblioteka umożliwiła sprawdzenie dnia tygodnia bazując na rzeczywistej dacie.
- Datetime – biblioteka umożliwiająca pobranie aktualnej daty, korzysta z niej większość funkcji, gdyż oczekuje się od programu działania w czasie rzeczywistym
- Json – biblioteka potrzebna do pracy z plikami imitującymi bazy danych, kluczowa w procesie wszelakich odczytów danych lub zapisu
- io – biblioteka służąca do imitacji plików (takich jak: historia rezerwacji lub cennik) w procesie testowania programu

Proces uruchomienia:

Jako, że program nie posiada graficznego interfejsu użytkownika oraz korzysta z wbudowanych bibliotek nie będzie potrzeba instalacji dodatkowych plików, wszystkie pliki źródłowe są zawarte w repozytorium na gitlabie wydziałowym https://gitlab-stud.elka.pw.edu.pl/bkosinsk/pipr-projekt-plywalnia. Plik rozpoczynający działanie interfejsu to interface.py, plik ten należy skompilować w wybranym środowisku programistycznym.

Testy:

Każdy plik źródłowy posiada testy jednostkowe zawarte w plikach test_{nazwa_pliku}.py, część programu została również przetestowana w sposób kompilacji poszczególnych funkcji.

Sam proces testowania był bardzo ważny, ponieważ pozwalał sprawdzić, które elementy programu potrzebują jeszcze dopracowania, co więcej po implementacji modyfikacji można było sprawdzić czy nie psują one wcześniejszych testów. Wnioski jakie nasuwają się po przeprowadzeniu tego procesu to: należy z góry ustalić jaki format danych jest przekazywany z funkcji do funkcji – jeżeli jest taka możliwość to nie ma warto zmieniać typu danej; zawsze należy sprawdzić jaki typ argumentów funkcji jest otrzymywany. Reasumując testy jednostkowe dobitnie przypominają o skrupulatności oraz konsekwencji pisania kodu źródłowego.

Opis funkcjonalności oraz implementacji:

1.Sposób przechowywania historii rezerwacji:

Historia rezerwacji powinna przechowywać następujące informacje: dany dzień, godzina rezerwacji, typ klienta, dostępność torów. Należy uwzględnić, że ostatni z elementów zależy od typu klienta, ponieważ szkółki pływackie zajmują domyślnie cały tor, natomiast jeden tor może posiadać więcej niż jedną rezerwację klientów indywidualnych.

Zatem można zawrzeć wszystkie informacje w pliku o formacie ".json". Przykładowa, uproszczona architektura danego pliku:

```
{
        <year>: {
                <month>: {
                        <month_day>: {
                                <line>: {
                                        <hour>: <availability>
                                        }
                                }
                        }
                }
        }
```

Przy czym <availability> pokazuje ilość wolnych miejsc na danym torze. Sposób tworzenia powyższego pliku będzie polegał na stworzeniu w przód 3 miesięcy od daty otworzenia programu.

2. Sposób przechowywania cenniku pływalni:

Cennik powinien być zróżnicowany, zatem musi posiadać informacje takie jak: typ klienta, wiek (dla klientów indywidualnych), dzień tygodnia, pora dnia, cena za godzinę, cena za godzinę.

W tym przypadku również można skorzystać z pliku o formacie ".json". Przykładowa, uproszczona architektura danego pliku:

```
{ <week_day>: {
        <hour>: {
                clients_types: {
                        individual: {
                                <type>: <price_for_hour>
                                }
                        groups: <price_for_hour>
                        }
                }
        }
}
```

3. Sposób przechowywania informacji o dochodzie w danym dniu:

```
{
        'individual type': {
                type: <income>,
                }
        'group': <income_for_groups>
        }
```

W pliku financial_report.py poza formatem przychodu zawierają się również funkcję, które mają za zadanie między innymi kreowanie historii przychodu, modyfikacja (w momencie dokonania rezerwacji), obliczanie łącznego przychodu czy wydruk gotowego raportu

4. Algorytm proponowania najbliższego możliwego terminu:

Dostępność danego terminu będzie zależeć od typu klienta. Dla szkółek pływackich dostępny będzie tylko cały wolny tor, a dla klientów indywidualnych zależy to od ilości wolnych torów, gdzie maksymalna ilość klientów indywidualnych na jednym torze to 5 osób.

Znając typ klienta, datę, godzinę oraz długość rezerwacji można sprawdzić czy dany tor jest dostępny o tej godzinie, jeżeli nie to należy sprawdzić następny tor. Jeżeli wszystkie tory są zajęte o tej godzinie należy powtórzyć proces, przechodząc do następnej godziny. Jeżeli jest to godzina zamknięcia to należy po niej przejść do następnego dnia oraz powtórzyć proces zaczynając od godziny otwarcia.

Dodatkowo dla grup pływackich należy sprawdzić przy każdej godzinie czy już inne szkółki nie zajmują 35% wszystkich torów.

5. Algorytm rezerwacji:

Algorytm proponowania najbliższego możliwego terminu wiąże się z algorytmem rezerwacji, ponieważ po znalezieniu dostępnego toru można go zarezerwować - zmienić wartość <dostępność> w konkretnym miejscu w historii rezerwacji, dla szkółek zmienić na wartość 0, a dla klientów indywidualnych zmniejszyć o 1.

Ponad to algorytm ten powinien zarezerwować termin tylko w wypadku zatwierdzenia otrzymania płatności oraz wcześniej wyświetlić koszt rezerwacji.

6. Raportu finansowy:

Tworzenie raportu finansowego będzie oparte na założeniu, że można go wykonać tylko pod koniec dnia. Będzie on zawierał następujące informacje:

- Ilość poszczególnych typów klientów
- Przychód za każdy typ klientów
- Łączny przychód w danym dniu

Przykładowy format:

```
Typ klienta                          Przychód
-----------------------------------------
Szkółki pływackie                      30.50
Indywidualni:
- normalny                             15.75
- ulgowy                               10.00
-----------------------------------------
Łączny przychód                        56.25
```

7.Działanie interfejsu:

Podczas pierwszego uruchomienia programu, automatycznie powinien się utworzyć plik, w którym będzie przechowywana historia rezerwacji (3 miesiące w przód). Należy również rozpatrzeć w jaki sposób będzie można dotrzeć do cennika – program zapyta o wpisanie ścieżki pliku, jeżeli takowy został utworzony wcześniej lub będzie można go utworzyć poprzez program.

Po tej operacji będzie można przejść do faktycznych funkcji interfejsu:

```
1: Make a reservation
2: Display price schedule
3: Modify price schedule
4: Financial report
5: Exit
```

Przy czym opcja pod numerem 4 jest możliwa tylko w momencie rozpoczęcia się ostatniej godziny otwarcia pływalni. Opcja trzecia nie została zrealizowana.

Wnioski:

Aktualny program zawiera niedociągnięcia - brak możliwości dodawania kolejnych dni (historia rezerwacji nie jest generowana w nieskończoność), brak możliwość śledzenia czy program został uruchomiony w trakcie jakiegoś dnia – aktualnie należy wprowadzić dane ręcznie, aby przejść przez poszczególne warunki (plik interface.py funkcja launching())

Ponad to sposób realizacji interfejsu w terminalu nie należy do najbardziej optymalnych, po pierwsze brakuje elementów graficznych, po drugie jesteśmy ograniczeni jako użytkownicy programu, ponieważ program nie pomaga nam w wprowadzaniu danych – brak wyszukiwarki plików, instrukcje warunkowe bazujące na dwóch ścieżkach wyboru.

Proces realizacji rozwiązania uświadomił mi jak ważna jest dekompozycja problemu – wprowadzanie wszelakich zmian w programie przy dużej ilości linijek kodu nie jest wygodne w momencie, kiedy należy wprowadzać te zmiany w kilku miejscach na raz. Zauważyłem również, że problemy na pierwszy rzut oka proste do rozwiązania mogą okazać się bardzo trudne - często, z braku doświadczenia oraz pewnych umiejętności musiałem pójść na kompromis oraz założyć pewne aspekty – na przykład w procesie tworzenia cennika najpierw należy stworzyć podstawową porę dnia aby móc tworzyć inne, bardziej specyficzne.

Oświadczam, że całość pracy wykonana na potrzeby tego projektu została wykonana samodzielnie.

Bartosz Kosiński

Nr albumu: 310752