Adam Moszczyński

Biletomat

Celem projektu było opracowanie funkcjonalnego programu do maszyny z biletami, działającego na zasadzie powszechnie obecnych biletomatów. Projekt został stworzony na podstawie niżej wymienionych założeń, które zostały zatwierdzone przez opiekuna projektu.

Założenia projektu:

1. Możliwość wyboru języka na początku każdej uruchomionej sesji programu.
    - **1.1.** Dostępne języki to (Polski, Angielski, Turecki)
2. Możliwość zakupu biletów kartonikowych.
    - **2.1.** Typy biletów kartonikowych: 20 minutowy, 75 minutowy, 24 godzinny oraz 72 godzinny
3. Możliwość zakupu biletów okresowych.
    - **3.1.** Typy biletów okresowych: miesięczny, 3 miesięczny oraz roczny
4. Możliwość sprawdzenia terminu ważności zakupionego biletu okresowego.
    - **4.1.** Termin ważności sprawdzany jest w specjalnej metodzie, która odczytuje datę zakupu biletu i na jej podstawie oblicza termin ważności.
5. Możliwość sprawdzenia dostępnych środków finansowych w bazie danych klientów
    - **5.1.** Środki finansowe są predefiniowane w bazie danych i ich początkowa wartość jest ustalana poza systemem biletomatu.
6. Możliwość sprawdzenia dostępnych biletów typu prepaid.
    - **6.1.** Bilety prepaid są predefiniowane w plikach Ticket_data oraz Customer_data i nie można ich zmienić bezpośrednio w biletomacie.
7. Możliwość zgłoszenia problemu z automatem.
    - **7.1.** Automat tworzy do pliku Error_logs.txt wpisy każdego problemu, który został wykryty przez program wraz ze stemplem czasowym.
    - **7.2.** Zgłoszenie problemu polega na wpisaniu przez użytkownika wiadomości opisującej problem oraz załączeniu do niej error logów do pliku Problem_report.txt.
8. Bazą danych która przechowuje informacje o klientach jest Customers_data.txt
    - **8.1.** W bazie danych przechowywane są (imię, nazwisko, id biletu oraz saldo) klienta.
    - **8.2.** Klient, żeby kupić bilet musi być w bazie danych i musi mieć wystarczające środki.
    - **8.3.** Jeżeli klient jeszcze nie ma biletu to jego wiersz zawiera 4 komórki, z czego id_biletu jest puste.
9. Plik settings.py zawiera metody w których konfiguruje się ścieżki do plików oraz ceny biletów.
10. Zakupy są typu prepaid, przy kupnie biletu z salda zostaje odjęta jego cena.

Klasy programu:

Program automatu posiada wbudowane oddzielne klasy każdego z dostępnych języków, odpowiednio (PL, EN, TR). Każda z tych klas zawiera metody do odpowiadających im zdarzeń, które jedynie wyświetlają tekst. Po wybraniu odpowiedniego języka w głównym pliku interface.py zostaje stworzony obiekt tego języka, na którym wszystko będzie operować aż do zakończenia sesji.

Klasa machine_customer, jest klasą na podstawie której tworzy obiekt każdego klienta. Posiada atrybuty takie jak: Imię, nazwisko, id biletów oraz dostępne saldo. W momencie pierwszego odczytania pliku Customer_data.txt na początku każdej sesji zostaje stworzona lista obiektów klientów.

Klasa machine_ticket jest klasą na podstawie której tworzy się obiekt każdego biletu. Posiada atrybuty takie jak id biletu, typ biletu, datę zakupu biletu. W momencie pierwszego odczytania pliku Ticket_data.txt na początku każdej sesji zostaje stworzona lista obiektów biletów.

Klasa error_classes jest plikiem w którym są wszystkie klasy błędów, jakie program powinien obsługiwać. Takie jak informacja o błędnym wyborze lub o niewystarczających środkach na zakup biletu.

Klasa system jest zainicjowana w pliku system.py. Posiada ona wszystkie niezbędne do prawidłowego działania całego programu metody. Metody w tej klasie odpowiadają za odczyt i zapis danych, sprawdzenie poprawności informacji, operacje na dacie oraz operacje opisane w założeniach projektu.

Interface jest to główny plik całego projektu. Współpracuje zarówno z klasami języków, jak i systemu. Jest to główne miejsce interakcji z użytkownikiem. Przyjmuje dane wejściowe wpisane przez użytkownika. Na podstawie tego co wskazał użytkownik wybiera odpowiednią opcję i zebrane dane wywołuje w metodach systemu. System po przetworzeniu zwraca lub zapisuje odpowiednią wartość, która ostatecznie jest przekazywana do danego obiektu z wybranym językiem. Po czym wywołana w obiekcie metoda interfejsu językowego wyświetla odpowiedni tekst. Całość jest w pętli, która za pomocą try oraz except reaguje adekwatnie do występującego błędu.

Instrukcja:

Uruchomienie aplikacji odbywa się przez podanie interpreterowi pliku Ticket_machine.py. Program będzie od razu gotowy do użycia. W terminalu powinny wyświetlić się opcje wyboru języka. Interakcja z programem odbywa się przez wpisywanie cyfry, która jest obok interesującej nas opcji. Po jej wpisaniu należy zatwierdzić wybór przyciskiem enter. Po zakończeniu polecenia, czyli np. sprawdzeniu środków na koncie. Program wróci do menu wyboru języka po 4 sekundach.

Program operuje na dwóch bazach danych. Pierwsza o nazwie Customer_data. W, której znajdują się dane klientów. Linia w Customer_data powinna zawierać id, first_name,

last_name, ticket_id, funds. Komórka ticket_id przechowuje id posiadanych biletów. Bilety oddzielone są średnikami. Wygląda to np. Tak ;1;2;5;6 . Podany użytkownik posiada 4 bilety o id 1, 2, 5, 6. Dane o tych biletach przechowywane są w Ticket_data.txt. W bazie danych biletów przechowywane jest ticket_id, ticket_type, ticket_date. Ticket _id jest unikatowym identyfikatorem każdego biletu, na podstawie czego określa się właściciela. Ticket_type jest typem biletu, który jest jednym z predefiniowanych (20min, 75min, 24h, 72h, 1m, 3m, 1y). Są to odpowiednio typy opisane w założeniach projektu. Bilety typu prepaid zapisane są w rubryce ticket_type jako kwota doładowania, czyli 3 to bilet 3zł itp.

Zgodnie z treścią polecenia, biletomat ma tylko sprawdzać bilety typu prepaid (ich zakup odbywa się poza biletomatem). Klienci Adam Moszczyński oraz Doralyn Dovermann posiadają zakupione bilety. Pierwszy z nich jest o wartości 10zł a drugi 26zł. W przypadku chęci sprawdzenia biletów prepaid dla innych osób, trzeba dodać je do bazy danych.

Refleksja:

Podsumowując całość prac nad projektem biletomatu, mogę śmiało stwierdzić, że zostały wykonane wszystkie cele otrzymane w zadaniu. Dodałem kilka dodatkowych funkcjonalności, żeby mój program jak najbardziej przypominał działanie prawdziwego biletomatu. Cele te zostały doprecyzowane skonsultowanymi z opiekunem założeniami. Zaczynając pracę nad projektem postawiłem sobie wymaganie, żeby żadna funkcja nie marnowała swojego potencjału i była użyteczna w wielu przypadkach. Dało to zadowalające efekty w późniejszych pracach, gdy przez błędne założenie niektóre rzeczy nie chciały działać. Zmieniając kilka wartości mogłem w stanie przywrócić je do stanu używalności. Oszczędziło mi to wielokrotnego zmieniania prawie całego kodu, jak i możliwości łatwej aktualizacji w przyszłości.

Zważywszy na to założenie, dużą trudność sprawiło mi wymyślenie łatwo edytowalnego interfejsu językowego, który pozwoli mi na szybkie przełączanie między językami oraz na niepowtarzanie komend. Jednak po wielu głębszych przemyśleniach udało mi się stworzyć uniwersalny interfejs, do którego dodając jeden plik z tłumaczeniem można dodać nowy język. Kolejnym problemem jaki pojawił się podczas tworzenia projektu były zapętlenia kołowe (circular loops). W początkowym założeniu uznałem, że będą trzy główne współpracujące ze sobą pliki, które będą wymieniać się informacjami. Jednak praktyka pokazała, że nie jest to najlepszy sposób. Przekształciłem wtedy początkową koncepcję. Zrobiłem jeden główny plik interfejs, który tworzył obiekty interfejsów językowych oraz systemu. Plik ten wywoływał wszystkie operacje wyświetlenia tekstu, kalkulacji z innych klas a sam w sobie pozwalał na interakcję z użytkownikiem.

W przypadku baz danych doszedłem do wniosku, że postawie na prostotę i łatwość w obsłudze. Dlatego stworzyłem je w formacie txt. Nie zajmują dużo miejsca i są dość intuicyjne w dodawaniu nowych danych zarówno przez jakiś zewnętrzny program jak i roboczo ręcznie. Jednak i tutaj pojawił się problem, jak zapisać dużą liczbę biletów dla jednego klienta. Zrobiłem więc dwie bazy danych, w których jedna zawiera kilentów z id biletów jakie posiadają, a druga posiada listę biletów. Za największe wyzwanie uznałem jednak współpracę

z pytest. Przez bardzo długi czas nie wykrywał testów do kluczowych plików projektu i opóźniał moją pracę.

Za największą zaletę swojego projektu uważam zdecydowanie sposób implementacji interfejsów językowych. Na większą uwagę zasługuje również modularność kodu, która pozwala na duże zmiany w kodzie za pomocą np. zmiany w jednej funkcji. Dość oczywistą rzeczą jest także to, że mój program działa, w każdym aspekcie jaki był w jego założeniach.