Zadanie z F# (lab 6 i 7):

I. Zadanie 6 (100%) – zaimplementować wszystkie zadania z instrukcji laboratoryjnej i pokazać działający skrypt i progra.

II. Zadanie 7 (100%) – zaimplementować prosty system biblioteczny **(100%)**. Program powinien być zaimplementowany jako aplikacja konsolowa z menu obsługiwanym z klawiatury.

Program powinien zawierać 2 rekordy:

- `DaneKontaktowe` – zawierający podstawowe dane takie jak: Imię, Nazwisko, data urodzenia, nr karty bibliotecznej, Adres email (opcjonalny).
- `Czytelnik` – zawierający numer identyfikacyjny czytelnika, rekord `DaneKontaktowe` opisujący czytelnika (opcjonalny), wysokość kaucji (depozytu) jako typ decymalny w USD, datę dołączenia oraz informację o statusie konta (Standard/Premium). Opis czytelnika (rekord `DaneKontaktowe`) jest opcjonalny.

Informacje o minimalnie czterech czytelnikach mogą być wczytywane z pliku JSON (nie mogą być zahardkodowane). Przynajmniej jeden z czytelników powinien mieć a przynajmniej jeden nie mieć, informacji opisujących (`DaneKontaktowe`). Z kolei, przynajmniej jedne informacje opisujące powinny posiadać a inne nie posiadać adres email.

Ponadto należy wykorzystywać **unię dyskryminowaną** określającą **stany niereprezentowalne**. Na przykład, rekord `DaneKontaktowe` musi zawierać numer telefonu *lub* adres pocztowy. Brak obydwu jest niedozwolony, aczkolwiek posiadanie obydwu jest poprawne.

Dla każdego czytelnika historia wypożyczeń przechowywana jest w osobnym pliku **json albo xml**. Plik powinien zawierać elementy powiązane z czytelnikami, w których skład wchodzą:

- numer identyfikacyjny czytelnika
- lista wypożyczeń. Wypożyczenie to **krotka** zawierająca informację o numerze ISBN książki oraz informację, czy książka została zwrócona.

Zdefiniować **funkcje** pozwalające na:

- `getLoanHistory` – pobranie listy wypożyczeń danego czytelnika.
- `isPatronForLongerThan` – sprawdzenie, czy dany czytelnik jest zapisany do biblioteki dłużej niż zadana liczba dni.
- `addFine` – zwiększenie salda kar na koncie czytelnika (kary są w PLN).

Program powinien zawierać **klasę `Library`** realizującą główną funkcjonalność (metody) systemu:

- awansowanie statusu czytelnika ze Standard na Premium, w przypadku kiedy czytelnik wypożyczył więcej niż (bazując na `getLoanHistory`) pewną ustaloną liczbę

książek oraz czy jest czytelnikiem dłużej niż (bazując na `isPatronForLongerThan`) pewien określony termin.
- pobranie informacji o wybranym czytelniku.

GUI w konsoli (np. przy użyciu Terminal.Gui lub proste `printfn`/`Console.ReadLine`) powinno pozwalać na wykonanie następujących operacji:

- odczytanie informacji o czytelniku: Imię, Nazwisko, Email.
- dodanie kary (w PLN) do konta czytelnika.
- sprawdzenie, czy suma naliczonych kar (PLN) nie przekracza wartości kaucji (USD) (należy pamiętać o innych walutach).
-

Należy zdefiniować jednostki miary EUR i PLN oraz relację wymiany pomiędzy nimi. Współczynnik wymiany należy pobrać z Internetu, np. serwis: https://stooq.pl/q/?s=eurpln i https://stooq.pl/q/?s=plneur.

Proszę wygenerować, odpowiednio wyedytować (usuwając niepubliczne składniki) i dołączyć do projektu biblioteki F# plik sygnatur (*.fsi).

Artykuły:

- Monads for functional programming

Linki:

- Map and Bind and Apply, Oh my!" series
- F# Domain Model Validation with Active Patterns
- LUHN validation with F#
- FsCheck: Random Testing for .NET
- Walidacja PESEL w PHP