Zadania 2

2.1

Zamień wszystkie asercje w pl.sda.doctor.PatientServiceTest na ich odpowiedniki z AssertJ

2.2

Klasa `pl.sda.db.UserDatabase` ma metody:

- `public UserDatabase(Clock clock)` – konstruktor przyjmuje obiekt klasy Clock (przydatne przy testowaniu)
- `registerUser(User user)` - metoda do rejestracji użytkownika, dodaje go do listy użytkowników, ale tylko jeśli:
  - `user != null` (jeśli jest = null wyrzuca `NullPointerException`)
  - imie użytkownika jest dłuższe niż 3 znaki
  - data urodzenia jest po dzisiaj
  - długość hasła przekracza 7 znaków
  - długość PESEL jest równa 11
  - PESEL użytkownika nie jest już w bazie
    Jeśli któraś z walidacji nie przejdzie wyrzucany jest odpowiedni wyjątek (zajrzyj do implementacji metody)
- `List<User> findUsersByNameContaining(String name)` – metoda zwracająca wszystkich użytkowników mających w imieniu przekazany do metody argument
- `List<User> findUsersWithPersonalPage()` – zwraca jedynie użytkowników, których personalPage nie jest równe null
- `Optional<User> findUserWithPesel(String pesel)` - szuka użytkownika po numerze PESEL, jeśli znajdzie to zwraca użytkownika w Optionalu, jeśli nie znajdzie to zwróci Optional.empty()
- `public List<User> findUsersWithBirthDateBetween(LocalDate from, LocalDate to)` – zwraca wszystkich użytkowników, którzy mają daty urodzin pomiędzy podanymi w agumentach datami (włącznie)
- `public List<User> findAll()` – zwraca wszystkich użytkowników, możesz jej użyć, żeby sprawdzić czy wybrany użytkownik został dodany lub usunięty
- `public User login(String name, String password)` – próbuje zalogować użytkownika podaną nazwą i hasłem, jeśli to się nie powiedzie wyrzuca `UserNotFoundException`

a) Napisz testy jednostkowe do wszystkich wymienionych wyżej metod używając AssertJ i wszystkich funkcji bibliotek do testowania poznanych

wcześniej, pamiętaj o braniu pod uwagę przypadków brzegowych i testowaniu wyjątków

b) Dopisz metodę `void removeUser(String pesel)` przyjmującą numer PESEL w argumencie i pozwalającą usuwać użytkownika o podanym numerze PESEL
c) Dopisz do `pl.sda.db.UserDatabase` metodę `List<User> findUsersWithBirthdayToday()`, która powinna pobierać dzisiejszy dzień używając `LocalDate.now(clock)`, a następnie przefiltrowywać użytkowników sprawdzając, którzy z nich mają dzisiaj urodziny, dopisz testy
d) Dopisz metodę sprawdzającą, kto z użytkowników podał nieprawidłową datę urodzenia w zestawieniu z numerem PESEL, przyjmujemy że 6 pierwszych cyfr numeru PESEL to YYMMDD gdzie YY to dwie ostatnie cyfry roku MM to miesiąc a DD to dzień, sygnatura metody to: `List<User> findUsersWithInvalidPeselAndBirthDate()`
e) *** napisz metodę `Map<Integer, List<User>> findAllGroupedByYearBorn()`, zwracającą mapę zawierającą w kluczu rok urodzenia, a w wartości użytkownika, który urodził się w danym roku, np.
   Jan Kowalski urodził się 01.01.1993
   Anna Kowalska urodziła się 02.06.1993
   Paweł Nowak urodził się 02.04.2001
   W rezultacie funkcja dla powyższych danych wejściowych powinna zwrócić mapę z dwoma kluczami o następującej zawartości:
   - Klucz 1993 – lista obiektów User zawierająca Jan Kowalski i Anna Kowalska
   - Klucz 2001 – lista obiektów User zawierająca Paweł Nowak

   Wskazówki:
   f) Niektóre przydatne metody z AssertJ:
      a. assertThat(<Optional>).isNotEmpty(); <- sprawdza czy Optional podany w argumencie jest nie pusty
      b. assertThat(<Lista>).isNotEmpty(); <- sprawdza czy Lista podana w argumencie nie jest pusta
   g) W klasie testowej (pl.sda.db.UserDatabaseTest) stwórz funkcję `public void setup()`, oznacz ją adnotacją `@Before` i przygotuj w niej dane, które zostaną użyte w trakcie testów do testowania logowania i funkcji znajdujących wybranych użytkowników (a więc zarejestruj kilku użytkowników)