Dokumentacja końcowa projektu PROI, Tomasz Foryś 331373, Temat:przychodnia lekarska

Założenia projektowe:
Celem projektu było stworzenie symulatora przychodni lekarskiej, składającej się z pacjentów, lekarzy, gabinetów, recepcji, kolejki do gabinetów. Dodatkowo zaimplementowałem prostą klasę dolegliwości pacjentów, ze zróżnicowanym czasem leczenia oraz klasę samej kliniki odpowiadającej za zakończenie programu. W celu odtworzenia równoczesnych akcji lekarzy użyłem threadów i biblioteki mutex, a w celu zapisywania i wypisywania danych korzystałem z biblioteki spdlog.

Odpalanie programu:
W głównym folderze projektu (https://gitlab-stud.elka.pw.edu.pl/tforys/proi_projekt) znajduje się plik executable main, będąc w tym samym folderze wystarczy wpisac ./main w terminalu, żeby odpalić projekt. W razie jakichś zmian należy wejść do folderu build i stworzyć nowy plik executable main za pomocą komendy make, później ponownie wystarczy go odpalić za pomocą komendy ./main .
W folderze głównym można znaleźć folder “logfiles”, gdzie znajdują się pliki:
-assign.log - zawierający logi dotyczące przydzielania pacjentów do doktorów
-debug.log - logi zawierające dokładne choroby i czasy ich wykonywania pacjentów
-finish.log - logi dotyczące leczenie pacjentów
-queue.log - logi dotyczące dodawania pacjentów do kolejki
-all.log - logi zawierające widok z terminala

Opis działania symulatora:
Symulator opiera się na jednoczesnym działaniu threadów recepcji, gabinetów oraz kliniki.

Klinika:
Klinika zawiera zmienne:
-workingReceptionists -ilość pracujących recepcjonistów w danym momencie
-open - wartość prawda/fałsz, czy klinika jest otwarta
Klinika ciągle sprawdza, czy liczba pracujących recepcjonistów jest większa od 0, jeśli nie, pokazuje komunikat o zaprzestaniu przyjmowania pacjentów, po czym zmienia flage open na Fałsz.

Recepcja:
Recepcja zawiera zmienne:
-name - imie recepcjonisty
-timeToGetPatients - czas dodanie jednego pacjenta do kolejki
-patientsToServe - ilość pacjentów, która zostanie przyjęta przez danego recepcjonistę
Recepcja przyjmuje daną ilość pacjentów we wskazanych odstępach czasowych, dodając ich do kolejki. Przykładowo podane są dwie recepcjonistki, jedna, “Mary”, przyjmuje 5 pacjentów, każdy co pół sekundy, druga “Clara” przyjmuje 10 pacjentów co 0.1 sekundy. W momencie, kiedy recepcjonista przyjmie wszystkich swoich pacjentów, zatrzymuje swoją pracę i daje znać klinice, że liczba pracującej recepcji się zmniejszyła.

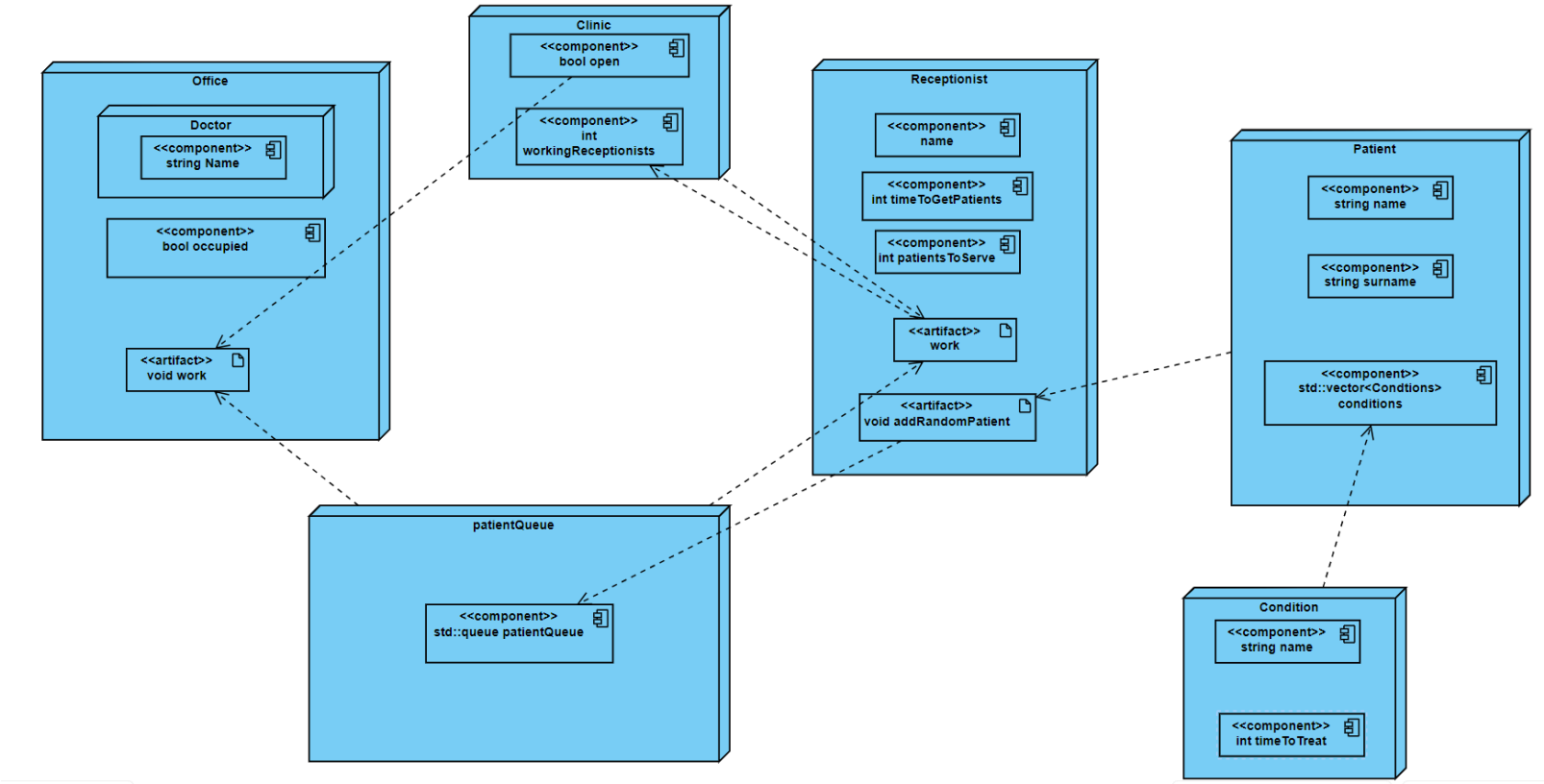
Dokumentacja końcowa projektu PROI, Tomasz Foryś 331373, Temat:przychodnia lekarska

Gabinet:
Gabinet zawiera zmienne:
-currentDoctor - Doktor obsługujący dany gabinet
-currentPatient - Pacjent obecnie znajdujący się w gabinecie
-occupied - bool czy ktoś znajduje się w gabinecie
Gabinet odpowiada za pobieranie pacjentów z kolejki, przekazywanie ich doktorom i odesłanie ich, jeśli skończyli swoją pracę.

Dodatkowe klasy:
Doctor:
-name -imie doktora
Doktor odpowiada za zajmowanie się pacjentami, ma metodę treatPatient, która analizuje symptomy danego pacjenta i działa zgodnie z nimi.

Condition:
-name - nazwa dolegliwości
-timeToTreat - czas w ms ile zajmuje zajęcie się dolegliwością
Klasa Condition jest klasą bazową dla 10 innych klas takich jak Headache, StomachAche, Fracture.

Poniżej znajduje się uproszczony diagram interakcji klas:



Wnioski i refleksje:
Uważam, że ten projekt był ciekawą okazją do nauki o multithreadingu i tym jak go bezpiecznie wykonywać. Myślę, że na pewno ta wiedza mi się przyda przy tworzeniu osobistych projektów. Spdlog również okazał się być ciekawym zasobem z wysokim poziomem customizacji. Niestety nie wiedziałem jakie testy powinny być do tego symulatora napisane, biorąc pod uwagę jego losową naturę i fakt, że czas wykonywania nie jest nigdy stały. Myślę, że gdyby było więcej osób w zespole, to na pewno możnaby by ten symulator dalej rozwinąć.