AKADEMIA GÓRNICZO-HUTNICZA

Dokumentacja do projektu

47. Patient and Doctor Scheduler

z przedmiotu

Języki programowania obiektowego

Elektronika i Telekomunikacja PL, III rok

Artur Nowak

grupa poniedziałek 19:25

prowadzący: Rafał Frączek

25.01.2020

1. Opis projektu

Celem projektu było stworzenie klas “pacjent” i “lekarz” pozwalających stworzyć system obsługi pacjentów przez lekarzy w którym lekarz nie może przyjąć więcej niż 16 pacjentów w ciągu dnia.

2. Project description

Projects main objective was creating classes “patient” and “doctor” allowing creation of appointments system such as doctors are not allowed making more than sixteen appointments with patients during the day.

3. Instrukcja użytkownika

Zaraz po uruchomieniu projektu widzimy panel główny z sześcioma opcjami. Opcje te są następujące:

1. Patients data and appointments - w tej opcji wyświetlane są dane wszystkich pacjentów i ich najbliższe wizyty

2. Doctors data - w tej opcji wyświetlane są dane wszystkich lekarzy

3. Doctors calendar - w tej opcji wyświetlane są dane wszystkich wizyt

4. Add appointment - w tej opcji możemy dodać wizytę do kalendarza

5. Add Patient - w tej opcji możemy dodać pacjenta

6. Add Doctor - w tej opcji możemy dodać lekarza

Jeśli podana zostanie cyfra inna niż z przedziału jeden do sześć program zakończy swoje działanie.

4. Kompilacja

Do obsługi programu wystarczy standardowa kompilacja.

5. Pliki źródłowe

Całość projektu znajduje się w jednym pliku źródłowym typu “.cpp”. Plik zawiera w sobie klasy “Doctor” i jej klasę pochodną “Appointment” oraz klasę “Patient”. Dodane zostały przykładowe obiekty umożliwiające weryfikację poprawności działania wyświetlania, dodawania oraz limitu szesnastu pacjentów na dzień.

6. Zależności

Brak

7. Opis klas

W projekcie utworzono następujące klasy:

- Doctor - reprezentuje lekarza w naszym systemie obsługi pacjentów. Lekarz identyfikowany jest za pomocą imienia, nazwiska, numeru PESEL, numeru PWZ i specjalizacji. Do tej klasy utworzono klasę pochodną Appointment. W klasie znajdują się standardowo “gettery” i “settery” oraz metoda “stringto” wyświetlająca wszystkie dane jednocześnie.
- Appointment - reprezentuje wizytę w naszym systemie obsługi pacjentów. Jest klasą pochodną klasy Doctor i rozszerza ją o dzień, miesiąc, rok, godzinę i minuty wizyty. W klasie znajdują się standardowo “gettery”, “settery”,metoda “stringto” wyświetlająca wszystkie dane jednocześnie oraz niezbędne przeciążenia operatorów umożliwiające dodanie obiektów z tej klasy do map.
- Patient - reprezentuje pacjenta w naszym systemie obsługi pacjentów. Pacjent identyfikowany jest za pomocą imienia, nazwiska, adresu i numeru PESEL. W klasie znajdują się standardowo “gettery” i “settery” oraz metoda “stringto” wyświetlająca wszystkie dane jednocześnie.

8. Zasoby

Brak

9. Dalszy rozwój i ulepszenia

Projekt w przyszłości można by ulepszyć o kilka następujących funkcjonalności:

- Usuwanie lekarzy, pacjentów i wizyt - w projekcie brakuje możliwości usuwania obiektów z klas “Doctor”, “Appointment” i “Patient”.
- Symultaniczne dodawanie pacjentów i wizyt - tworząc wizytę podajemy dane pacjenta, lekarza i dane wizyty. Jednakże podanie danych pacjenta w trakcie tworzenia wizyty nie powoduje dodania pacjenta do rejestru znajdującego się w opcji pierwszej panelu głównego.
- Stworzenie ograniczeń danych - w projekcie brakuje ograniczeń formatu podawanych danych, na przykład nic nie stoi na przeszkodzie podania numeru PESEL składającego z jednej cyfry.
- Stworzenie prawdziwego kalendarza - w projekcie nie ma ograniczeń co do podawania dat kalendarzowych i godzinowych, na przykład podanie terminu wizyty o 25:99 dnia 99.01.2099 jest możliwe. Wizyty również nie są w żaden sposób sortowane względem daty.
- Poprawa przejrzystości - w celu poprawienia przejrzystości obiekty, metody i funkcje powinny zostać zadeklarowane w plikach “header’owych” w celu poprawy przejrzystości.
- Działanie mapowanie - do projektu wprowadzono mapowanie obiektów klasy “Appointment” na obiekty klasy “Patient”. Rozwiązanie to przysparza wielu problemów na przykład przy próbie przeszukiwania takiej mapy. Należałoby znaleźć lepsze, unikające niepotrzebnych przeszukiwań mapy.

10. Inne

Brak