Introduction

Šio kursinio darbo tikslas yra sukurti ligoninės valdymo sistemą, kuri leistų tvarkyti medicinos personalą (gydytojus, slaugytojus) ir pacientus. Sistema turi galimybę pridėti, pašalinti, rodyti medicinos personalą ir pacientus bei įkelti duomenis iš failo. Mano programa yra ligoninės valdymo sistema, skirta tvarkyti gydytojų, slaugytojų ir pacientų duomenis bei jų veiklą. Paleisti programą galima per bet kurį Python aplinkos konsolės interfeisą. Pakanka atsidaryti komandinę eilutę ir įvykdyti programos failą. Programa turi meniu, kuriame yra pasirinkimai pridėti arba pašalinti gydytojus, slaugytojus, pacientus, bei spausdinti medicinos personalą arba pacientų sąrašą. Naudotojui tiesiog reikia pasirinkti norimą veiksmą ir sekti nurodymus konsolėje.

Body/Analysis

Polymorphism:

Polimorfizmas pasireiškia, kai šablonas (pvz., MedicalStaff) leidžia naudoti daugelį išvestinių klasių objektų (pvz., Doctor ir Nurse) taip, kad šie objektai gali būti naudojami vietoje pagrindinės klasės objekto. Pavyzdžiui, Doctor ir Nurse klasės paveldi MedicalStaff klasę ir perrašo jos metodą action, tačiau šių klasės objektai gali būti naudojami toje pačioje funkcijoje ar metode.

Abstraction:

Abstrakcija yra panaudota MedicalStaff klasės pavidalu, kuri apibrėžia bendrą elgseną medicinos personalui, bet nepateikia konkretių įgyvendinimo detalių. Taip pat, HospitalManager klasė naudoja abstrakciją valdydama medicinos personalą ir pacientus be reikalavimo dėl jų specifinių savybių ar elgsenos.

Inheritance:

Paveldėjimas yra panaudotas Doctor ir Nurse klasėse, kurios paveldi bendrą elgseną iš MedicalStaff klasės. Tai leidžia dalinai atstatyti kodo naudojimąsi jau esamomis klasėmis ir išvengti dubliuoto kodo rašymo.

Encapsulation:

Uždarumas pasireiškia panaudojant privačias klasės savybes ir metodų prieigą. Pavyzdžiui, klasėse Doctor ir Nurse, name yra privati savybė, prieinama tik per viešus metodus (getter'ius ir setter'ius), todėl užtikrinama, kad jos negalima tiesiogiai pakeisti iš išorės.

Taip pat HospitalManager klasė panaudoja uždarumą, nes jos savybės ir metodai yra saugomi privačiai, o prieiga prie jų yra kontroliuojama per viešus metodus.

Singleton:

HospitalManager klasė yra įgyvendinta kaip vienintelis egzempliorius, naudojant Singleton šabloną. Tai užtikrina, kad ši klasė turi tik vieną egzempliorių per visą programą, kuriuo galima dalintis visur, ir užkerta kelią nekontroliuotam naujų egzempliorių kūrimui.

Decorator:

Dekoratoriaus šablonas panaudotas decorator funkcijoje, kuri dekoruoja Doctor ir Nurse klasės metodus, leidžiant pridėti papildomą funkcionalumą (pvz., atspausdinant pranešimus prieš ir po action metodo iškvietimo). Taip suteikiamas papildomas funkcionalumas be jų pagrindinių metodų keitimo.

Rezultatai

Ligoninės valdymo sistemos įgyvendinimas vyko sklandžiai, visos pagrindinės funkcijos veikia kaip tikėtasi. Problema atsiranda, kai pradedu naudoti unittestus. Paleisdamas atskirą failą skirta unittestams visa informacija, kuri randasi tekstiniame faile dingsta.

Išvados

Šis kursinis darbas pasiekė ligoninės valdymo sistemos įgyvendinimą, kuris atitinka objektinio programavimo principus ir kūrimo šablonus. Programa teikia sprendimą medicinos personalo ir pacientų valdymui ligoninės aplinkoje. Būsimi programos perspektyvos gali apimti tolesnius tobulinimus, tokius kaip papildomų funkcijų pridėjimas medicinos personalo tvarkaraščiams, integracija su išorinėmis duomenų bazėmis ir vartotojo sąsajos tobulinimas geresnei naudotojo patirties užtikrinimui.

Resources

<https://www.dataquest.io/blog/unit-tests-python/>

<https://refactoring.guru/design-patterns/python>

<https://www.youtube.com/watch?v=5-uFpAPsBcQ&list=PL3EjD-tUP5iaVwj-I5689MhrV_x6LTfaP>

<https://www.youtube.com/watch?v=RwcEQb3CRps>

<https://www.youtube.com/watch?v=huaBJrfKv1c>