SOLID principi

Single responsibility principle

- Princip pojedinačne odgovornosti smo realizovali tako što smo za svaku postojeću klasu razdvojili metode koje se isključivo tiču atributa i cjelokupne djelatnosti samo te klase. Čime smo ostvarili željeno razdvanjanje poslova.
- Ostale metode smo odvojili u interface-e kao npr. metodu prikažiPlanner koja može zavisiti od budućih modifikacija na klasi Termin; te napimjer metodu napraviLoyalKarticu koja će ovisiti od budućih promjena na klasi LoyalKartica.

Open - Closed principle

- Ovaj princip smo ispoštovali tako što smo se direktno nadovezali na prethodni princip poštovanjem pojedinačne odgovornosti, te shodno tome osigurali da su atributi svih klasa napravljeni kao dobar temelj za buduće nadogranje, a samim time i obezbijedili njihovu cjelovitost (nema promjena nad njima).
- Primjer poštovanja ovog principa jeste urađen interface Pregledanje u koji se može dodavati novi način pregledanja (nezavisan od postojećih) u slučaju potrebe pravljenja dodatnih klasa za funkcionalnosti koje će podržavati pregledanje.

Liskov substitution principle

- Radi jednostavnosti ovog principa, specifično za našu aplikaciju, dodali smo apstraktnu klasu Korisnik koja povezuje čitav sistem i koju druge dvije klase Pacijent i Doktor nasljeđuju (iz ovih klasa se dalje sve nastavlja).
- Kao što Liskov princip zahtjeva obje klase, i Pacijent i Doktor su zamjenjive sa klasom Korisnik. U slučaju dodavanja novih klasa koje bi nasljeđivale klasu Korisnik, Liskov princip ne bi bio narušen jer po samom konceptu aplikacije, za novu klasu (odnosno novu vrstu korisnika) je nužno da posjeduje atribute klase Korisnik te njene metode, pri tome bi princip i dalje bio zadovoljen.

Interface Segregation Principle

- Sve metode koje nisu esencijalne za korisnika su, u skladu sa poštovanjem ostalih principa, prebačene u interface-e.
- Na navedeni način smo izbjegli postojanje „debelih“ klasa. Te smo eventualna ažuriranja istih značajno olakšali.
- Klase odvojene u interface se generalno manje koriste. Što je u skladu sa ovim principom.

Dependency Inversion Principle

- Kako ne treba ovisiti od konkretnih klasa tj. nasljeđivanja treba razmotriti slučaj da je osnovna klasa apstraktna. Ovaj princip je ispoštovan na primjeru apstraktne klase Korisnik, u slučaju dodavanja izvedenih klasa klase Doktor (naprimjer glavni doktor u ordinaciji, ili tehničar i sl.), osnovna klasa Korisnik ne podliježe promjenama. Samim tim je sistem agilniji za promjene.