SOLID PRINCIPI

S:

Klase kao jedinke bi trebale imati samo jedan razlog za promjenu, treba da znaju samo o jednoj stvari, trebaju imati pojedinačnu odgovornost.

Klasa Korisnik je klasa iz koje su izvedene klase Doktor i Pacijent i ona se brine o osnovnim podacima koji se tiču medicinskog osoblja i pacijenata. Zatim, u *klasi Doktor* imamo osnovne kontruktore, gettere i setter i tri najosnovnije aktivnosti koje doktor obavlja a to su azuriranje termina rasporeda, izdavanje recepata i nalaza i za koje mislimo da nema smisla razdvajati u posebne klase. *Klasa Pacijent* pored konstruktora, gettera i settera moze vrsiti osnovnu aktivnost a to je zakazivanje termina.

Klase Recept, Nalaz i MedicinskiKarton su zaduzene za osnovnu aktivnost a to je ispunjavanje odgovarajucih podataka.

Klasa Raspored je klasa iz koje su izvedene klase RasporedPregleda i RasporedZaZakazivanje i sadrzi gettere i settere. *Klasa RasporedZaZakazivanje* je odgovorna samo za aktivnosti koje se tiču termina, dok *klasa RasporedPregleda* brine o postavljanju statusa datog termina (slobodan, zauzet, otkazan).

Klasa Pregled se sastoji od tri osnovne komponente koje su kljucne da bi svaki pregled bio uspjesno zavrsen, a to je ispunjavanje medicinsko kartona, nalaza i recepta.

O:

Klase, moduli i funkcije trebali bi biti otvoreni za nadogradnji, ali zatvoreni za modifikacije.

Ukoliko želimo dodati nove metode takva akcija neće zahtijevati uređivanje već postojeće klase i njenih atributa. To pokazuje i cinjenica da klase međusobno koriste druge klase, tako da se desava samo "nadogradnja" i ponuda novih mogućnosti.

L:

Podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima.

Apstraktne klase u našem slučaju su dvije i to Korisnik i Raspored, one sadrže osnovne podatke bez kojih ove dvije klase ne bi mogle postojati kao samostalne, tako da se na svim mjestima gdje se koristi izvedeni objekat moze koristiti i osnovni objekat, a da pri tom takav nacin koristenja ima smisla.

I:

Klijenti ne treba da ovise o metodama koje neće upotrebljavati.

Mislimo da je nakon detaljne analize S dijela u kojem su navedene i osnovne aktivnosti svake od klasa, zadovoljen i ovaj princip jer svaka klasa "obavlja" upravo one aktivnosti (preko metoda) koje su najbitnije za uspjesan rad sistema. Možemo reći da nijedna od klasa u našem sistemu nije tzv. *debela klasa.* Za sada jos uvijek nismo definisali niti jedan interfejs, tj. nismo vidjeli potrebu za njim, pa ako Vi smatrate da imamo prostora za interfejs slobodno nam se obratite.

D:

Princip inverzije ovisnosti: Ne treba ovisiti od konkretnih klasa. Prilikom nasljeđivanja treba razmotriti slučaj da je osnovna klasa apstraktna.

Dakle, imamo da ovaj princip zahtijeva da pri naslijeđivanju od strane više klasa bazna klasa bude uvijek apstraktna, što je upravo kod nas i slučaj, klasa Korisnik i klasa Raspored jesu apstraktne klase.