

## STRUKTURALNI PATERNI

U sistem smo dodali *Bridge* i *Proxy patern*, a zatim su navedene situacije u kojima bismo mogli koristiti i ostale strukturalne paterne.

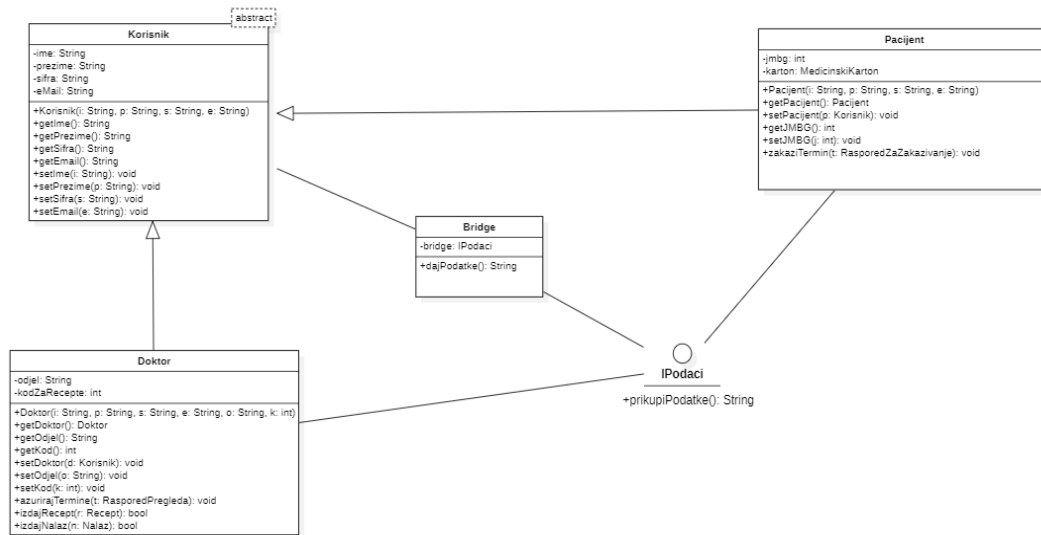
- **BRIDGE PATTERN**

Ovaj patern u praksi omogućava da se iste operacije primjenjuju nad različitim podklasama, kako bi izbjegli kreiranje novih metoda za postojeće funkcionalnosti.

U našem slučaju imali bismo sljedeću situaciju:

Korisnik želi da na jednom mjestu (putem jedne metode) čuva informaciju o ličnim podacima, pri tom Korisnik može biti Doktor ili Pacijent. Ono po čemu se ova dva korisnika razlikuju jeste *jmbg* koji je karakteristična oznaka za Pacijenta i *odjel* koji je osnovna oznaka Doktora.

U našem sistemu u okviru klase *Bridge* imamo atribut *bridge* tipa *IPodaci* i jednu metodu *dajPodatke()* u kojoj ćemo objediniti sve podatke koji se tiču specifičnog korisnika i koja zapravo poziva metodu implementacije u okviru interfejsa *IPodaci* koja se razlikuje u zavisnosti koji je Korisnik u pitanju.



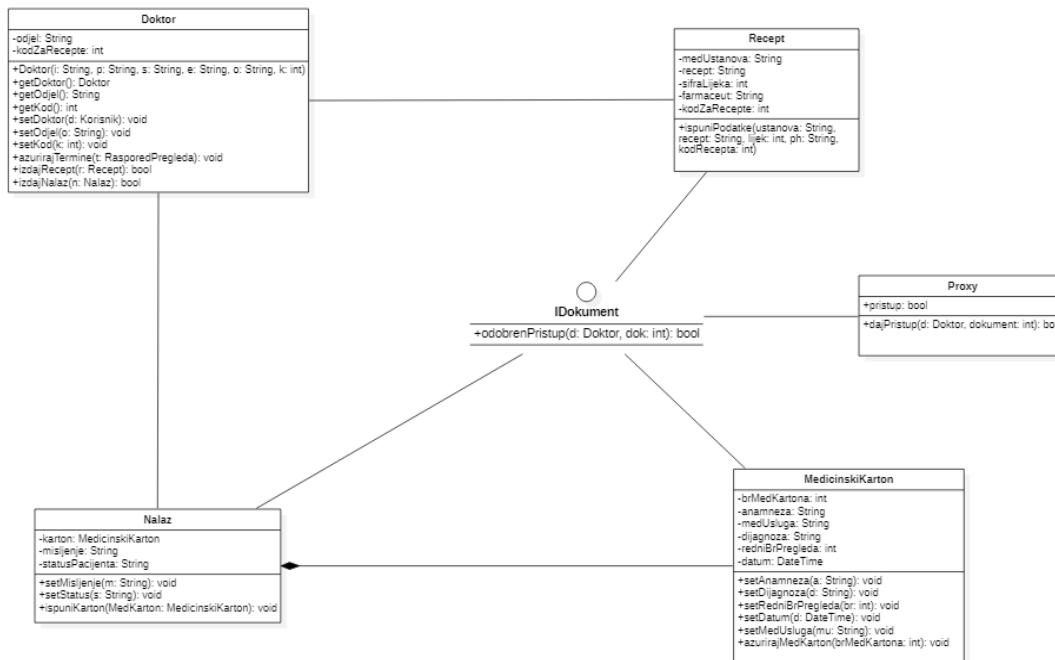
## • PROXY PATTERN

Uz pomoc Proxy paterna vršimo zaštitu pristupa resursima i kontrolu pristupa stvarnim resursima, odnosno da se ostvari jedan nivo dodatne sigurnosti u sistemu.

Situacija pogodna za ovaj paterna u našem sistemu:

Doktor želi da ograniči pravo pristupa Nalazu, Receptu i MedicinskomKartonu, odnosno želimo osigurati da samo doktor možemo ažurirati, mijenjati i dodavati informacije unutar navedenih klasa.

Navedeno ćemo ostvariti tako što ćemo uvesti *Proxy klasu sa atributom pristup tipa bool i metodom dajPristup koja kao prvi parametar prima Doktora, a kao drugi parametar prima parametar "dokument" tipa int (1-Nalaz, 2-Recept, 3-MedicinskiKarton) kako bi doktor mogao odabrati dokument kojem želi pristupiti. Imamo također i interfejs IDokument sa metodom odobravanja pristupa dokumentu.*



## • ADAPTER PATTERN

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. U situacijama kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter pattern.

U našem sistemu Adapter pattern bismo mogli iskoristiti u situaciji kada bi nam pored pdf formata Nalaza, Recepta i MedicinskogKartona trebao i neki drugi format (.txt i slicno) iz nekog razloga. Preko adaptera bismo konvertovali jedan format u drugi kako bi bio ostvaren krajnji cilj.

Pattern bismo implementirali preko klase nazovimo je FormatAdapter koja bi kao atribut imala dokument u jednom formatu i ako je potrebno vršiti neke izmjene u tom dokumentu mogli bismo dodati i neke metode, zatim intefejsa nazovimo ga IFormat koji bi imao jednu metodu za konvertovanje u drugi format.

## • FACADE PATTERN

Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na

podсистeme (implementacija podсистema skrivena od korisnika). Koristi se kada želimo koristiti samo mali dio velikog i kompleksnog sistema.

U našem sistemu mogli bismo uvesti klasu DokumentiFasada koja bi objedinjavala tri dokumenta koja su usko povezana Recept, Nalaz i MedicinskiKarton čije je ispunjavanje neophodno da bi jedan pregled bio uspješno obavljen. Unutar klase DokumentiFasada mogli bismo imati metodu koja vrši ažuriranje ovih dokumenata, jer se manje više popunjavaju slični podaci, zatim bismo mogli imati metodu koja bi vršila download ili printanje sva tri dokumenta zajedno (arhiviranje dokumenata) i slično.

- **DEKORATER PATTERN**

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija.

- **COMPOSITE PATTERN**

Osnovna namjena ovog paterna je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju. Ovaj patern također omogućava pozivanje iste metode nad različitim objektima sa različitim implementacijama.

U našem sistemu ovaj patern bi bio ispoštovan u slučaju Kartoteke. Naime, medicinskom kartonu može pristupiti više korisnika (pacijent, doktori kod kojih se pacijent vodi i slično). U kartoteci se nalaze svi medicinski kartoni pacijenata, pri čemu se svakom kartonu može jedinstveno pristupiti putem unosa broja medicinskog kartona, što bi se moglo ostvariti preko jedne metode. Ukoliko dođe do brisanja profila pacijenta doći će do brisanja medicinskog kartona pacijenta, što također obavljamo preko jedne metode i slično.

Patern bismo implementirali preko interfejsa nazovimo ga IKarton koji bi imao jednu metodu traziKarton(brMK: int) koja bi vršila pretragu kartona na osnovu proslijeđenog parametra.

- **FLYWEIGHT PATTERN**

Postoje situacije u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase (glavno stanje) od dijela klase koji nije uvijek isti za sve određene objekte te klase (tzv. sporedno stanje). Osnovna namjena Flyweight paterna je upravo da se omogući da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje.

U našem sistemu ovaj patern bismo mogli iskoristiti u slučaju downloada ili printanja nekog od dokumenata (Recept, Nalaz ili MedicinskiKarton) gdje bismo imali pristup glavnom stanju (pdf format) dokumenta a u slučaju downloada (u kombinaciji sa Adapter paternom) bismo radi smanjenja veličine dokumenta mogli downloadati dokument u drugom formatu. Sličan princip bismo imali i u slučaju printanja dokumenta.