STRUKTURALNI PATERNI

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

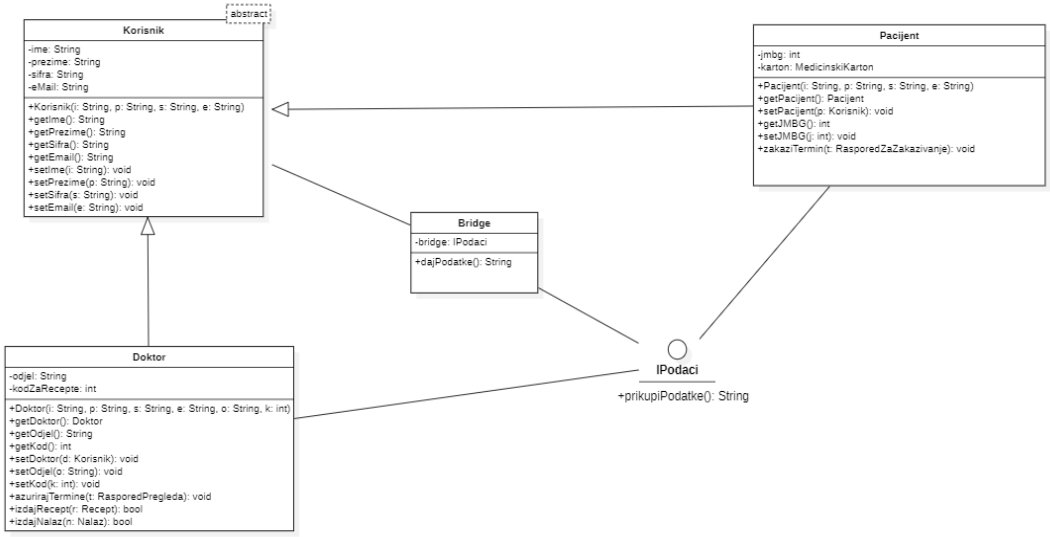
- **BRIDGE PATERN**

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

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 nasem sistemu u okviru *klase Bridge imamo atribut bridge tipa IPodaci i jednu metodu dajPodatke()* u kojoj cemo 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.



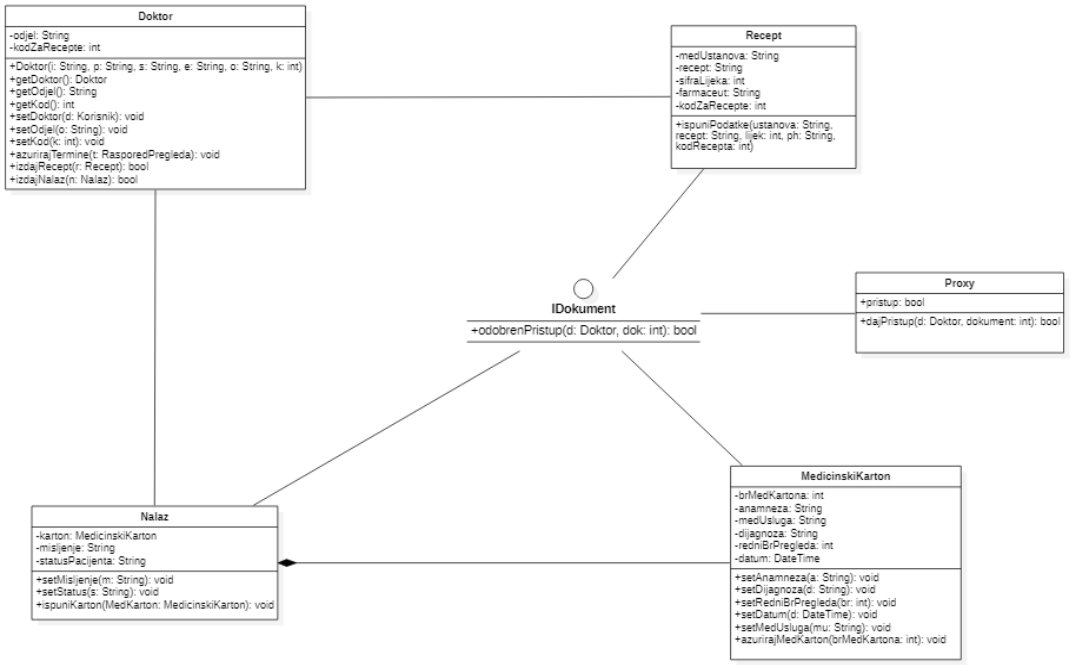
- **PROXY PATERN**

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 PATERN**
- **FACADE PATERN**
- **DEKORATER PATERN**
- **COMPOSITE PATERN**

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 moze jedistveno 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 PATERN**