Strukturalni paterni

1. **Adapter**
   U klasi „KorisnikSaNalogom“, trenutno imamo samo settere i gettere za korisnikovu listu rezervacija, mogli bi smo napraviti adapter koji bi nam umjesto liste vracao uredjen raspored rezervacija (*svaka kolona bi predstavljala jedan dan, interno bi bi bila implementirana kao matrica*). Pošto bi nam bilo bolje da onemogucimo pozivanje get motode na obicnu listu, tu bi nam pomogao „Class Adapter patern“, tj. da adapter naslijedimo iz klase „KorisnikSaNalogom“ i da implementira interfejs koji smo mi već napravili u dijagramu („*ITermini*“).

2. **Facade**
   Detaljno pregledajući cijeli sistem (*naš sistem i class dijagram su vrlo jednostavni i nema usko povezanih podsistema*), jedino mjesto gdje bi mogli iskoristiti ovaj strukturali patern je sa interfejsima „IMapa“ i „ITermini“ jer smo ih napravili da oba budu pomocni interfejsi nasim klasama „SportskiCentarSaTerminima“ i „SportskiCentar“, pa bi ih mogli smjestiti u jednu class Fasadu.

3. **Decorator**
   Kad bi htjeli iskoristiti ovaj strukturalni patern, mogli bismo dodati neke funkcionalnosti nasem administratoru koji vodi racuna o izgledu Sportova i Sportskih centara. Npr. mogli bismo mu dodati neke funkcije manipulacije sa slikom (*kao sto imamo u Wordu, npr. crop, fade, frame,...*).

4. **Bridge**
   U nasem sistemu postoji mogucnost placanja unaprijed rezervisanog termina. Buduci da korisnici naravno ne koriste iste banke, a znamo da je u svakoj banci uplacivanje na racun i skidanje novca sa racuna drugacije (*neke banke uzimaju vece naknade neke manje,...*), da bi postigli da u nasem sistemu svi korisnici budu osigurani da mogu koristiti sve svoje usluge banke, mogli bi napraviti interfejs „Bridge“ koji definira apstrakciju i ima razlicite implementacije posebno za svaku banku (*informaciju u kojoj je banci bi mogli dobiti iz broja racuna ili nekog drugog podatka koji je specifican za svaku banku*). Također smo primijetili da bi mogli iskoristiti ovaj strukturalni patern za prikaz Sportskog centra sa terminima i obicnog sportskog centra na isti nacin kako smo opisali. To smo takodjer dodali u nas class dijagram.

5. **Proxy**

Buduci da se najčešće ovaj strukturalni patern koristi za cuvanje podataka, neku autentifikaciju, dobili smo ideju da bi mogli u nasem sistemu implementirati klasu „AutentificationProxy“ koja nasljedjuje interfejs sa metodom za log in (*ona bi naravno primala korisnicko ime i lozinku*), koja bi odobravala ili odbijala pristupanje korisnickom racunu ukoliko neki podatak nije tacan, istu stvar bi mogli iskoristiti ukoliko bi se u sistemu trazilo da se upise lozinka ukoliko se zeli obrisati korisnicki racun ili otkazati rezervacija (*pošto znamo da je u nekim vecim aplikacijama ovo potrebno i vrlo dobro zamisljeno*)

6. **Composite**

Kada bi imali neki view na kojem bi zajedno prikazivali Sportove i Sportske centre na nekoj listi, pošto su to dva veoma razlicita objekta njihov prikaz bi bio realizovan na razlicit nacin. Iz tog razloga smo napravili interfejs „IPrikaz“ te klasu „ListaZaPrikaz“ koja u sebi drzi listu objekata tipa „IPrikaz“ te kao i Sport i SportskiCentar implementira istoimeni interfejs. Sada prilikom poziva metode „Prikazi“ za listu objekata „IPrikaz“ dobijati ce se odgovarajuca „Prikazi“ metoda.

Postoji još jedan hipotetski nacin na koji bi mogli iskoristiti ovaj patern. Pošto svaki veliki sportski centar ima odvojene sportove (stadion, kosarkasi teren, teniski teren, ...) svaki prikaz sportskog centra ce biti poseban za svaki sport (razne slike, druga lokacija, ...), time smo mi trenutno u sistemu dokazali da smo dobro odredili da povezanost izmedju Sporta i Sportskih centara bude kompozicija, jer brisanjem sporta automatski bi se obrisali i ti „dijelovi“ sportskih centara. Da smo u sistemu objedinili da jedan sportski centar moze odjednom davati usluge vise sportova i tako ih prikazivali tada bi nam dobro dosao ovaj strukutralni patern jer bi nam bilo u cilju da se posmatraju kao cjeline i Sport i Sportski centar. Mogli bi dodati jedan IComponent koji bi definirao interfejs operacije za nase objekte u kompoziciji. Tada bi se omogucilo editovanje (*uredjivanje, brisanje,...*) svih Sportova i Sportskih centara sa jednom metodom u IComponent.

7. **Flyweight**

Pošto se ovaj strukturalni patern koristi kada bi neke nase modelske klase imale neku osobinu koja se naknadno postavlja (*neobavezna je za rad same aplikacije*) te se toj osobini dodjeljuje neka default vrijednost koja se nalazi na samom sistemu radi smanjenog koristenja memorije. Pošto smo dizajnirali naš sistem tako da koristimo samo podatke (*atribute*) koji su osnovni i *potrebni* u aplikaciji, nemamo veliku potrebu za ovim paternom. Jedino gdje bi mogli iskoristiti ovaj patern je sa slikama u Sportovima i Sportskim centrima (*iako danas svaki sportski centar mora imati slike kako bi radio*), ako administrator ne

doda slike za odredjeni Sport ili Sportski centar da oni dobiju neku defaultnu vrijednost (*neke slike na nasem sistemu*).

U našem sistemu ćemo diretno implementirati i dodati na naš class dijagram Composite i Bridge paterne, vidimo i veliku primjernu na Proxy paternu ali njega ćemo implementirati ako kasnije u sistemu primijetimo veću primjenu za njega. Sam dizajn paterna koje ćemo dodati u naš sistem su graficki predstavljeni u posebnom fajlu našeg Class dijagrama.