Strukturalni paterni

Adapter patern

Adapter patern omogućava širu upotrebu već postojećih klasa. Ukoliko je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu koristi se Adapter patern koji kreira novu adapter klasu.

Adapter patern se može upotrijebiti kada korisnik želi dodati sliku, pri čemu se ne mora koristiti određeni format slike, već korisnici imaju izbor (.jpg, .png,...). Također, ovo se može primjeniti i za video, ukoliko korisnik odluči postaviti snimak pripreme jela (.mp4, .avi,...).

Facade patern

Facade patern osigurava više pogleda visokog nivoa na podsisteme (implementacija podsistema skrivena od korisnika).

Klasa *Recept* je fasada jer sadrži način pripreme i sastojke potrebne za recept, a koji se ne koriste izvan ove klase.

Decorator patern

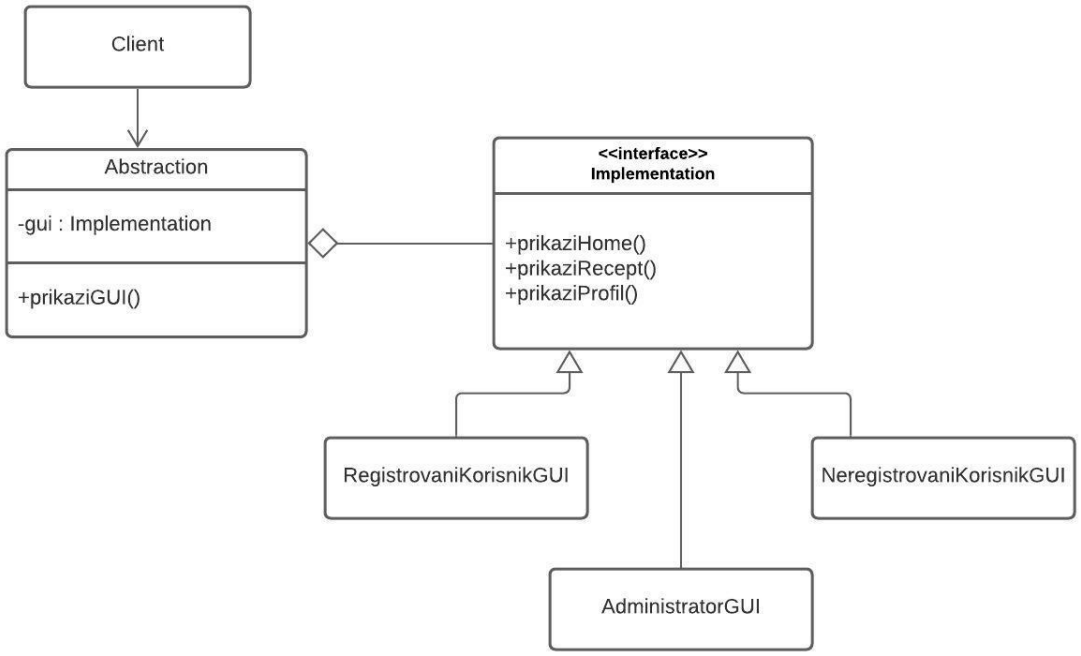
Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima. Objekat pri tome ne zna da je urađena dekoracija što je veoma korisno za ponovnu upotrebu komponenti softverskog sistema.

Decorator patern se može koristiti ukoliko korisnik želi urediti vlastitu sliku i/ili sliku recepta. Pod uređivanje slike se podrazumijeva, između ostalog: rotiranje, promjena veličine, promjena osvjetljenja i sl.

Bridge patern

Bridge patern omogućava odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije.

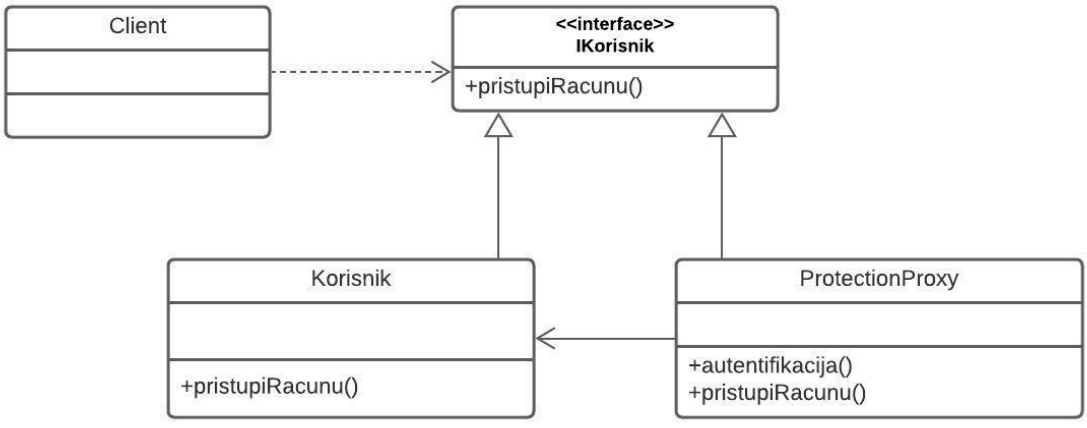
Kako u našoj aplikaciji treba pružiti različite GUI za registrovanog i neregistrovanog korisnika, te administratora, Bridge patern se može tu primjeniti.



Proxy patern

Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima.

Ovaj patern koristimo za kontrolu unosa lozinke prilikom prijave. U slučaju ispravno unesene lozinke, odobrava se pristup i korisnik će biti prijavljen na sistem, a u suprotnom korisnik dobija obavještenje o neispravnom unosu i mogućnost ponovnog unosa.



Composite patern

Da bi se mogao primjeniti ovaj patern potrebno je da se glavni dio aplikacije može predstaviti kao stablo. Međutim, u našoj aplikaciji to nije slučaj pa ovaj patern nećemo koristiti.
Recimo da smo imali apstraktnu klasu *Korisnik* i naslijeđene klase *RegistrovaniKorisnik* i *NeregistrovaniKorisnik*. Ako bismo željeli prikazati preporučene recepte za ove korisnike, mogli bismo prikazivati najnovije recepte za registrovane, a najbolje ocijenjene za neregistrovane korisnike. Za implementaciju ovog paterna trebali bi promijeniti klasu *Korisnik* u interfejs *IKorisnik* koja bi implementirala metodu *DajPreporuceneRecepte.*

Flyweight patern

Flyweight patern omogućava da više različitih objekata dijele isto glavno stanje, a imaju različito sporedno stanje.

Ovaj patern korisitimo kada sastojcima već upisanim u bazu mijenjamo atribute *kolicina* i *mjerna jedinica*.