Strukturalni paterni

1.Adapter patern

Ovaj patern koristimo kada je potreban drugačiji interfejs već postojeće klase, a ne želimo mijenjati postojeću klasu.

Možemo napraviti klasu Izvještaj. Ova klasa bi nam služila za kreiranje izvještaja o korisniku ili o nekom zahtjevu. Kako ne bismo morali pravit dvije različite klase izvještaja možemo implementirat adapter koji omogućava da ova klasa može kreirati i izvještaj o korisniku i o zahtjevu.

2.Facade patern

Ovaj patern se koristi da bi se korisnicima pojednostavilo korištenje složenog sistema koji se sastoji od više podsistema sa povezanim implementacijama.

Pomoću ovog paterna nešto komplikovano želimo pojednostaviti.

Klijent vidi samo krajnji objekat, dok je njegova struktura skrivena. Ovo omogućava da korisnik radi sa sistemom, bez da detaljno poznaje kako sistem funkcioniše. Facade patern u našem sistemu je primjenjen, s obzirom na to da korisnik vidi samo određeni dio sistema koji je njemu potreban. Korištenjem facade paterna se također smanjuje pojava grešaka pri rukovanju sistemom od strane korisnika.

Facade šablon je moguće primjeniti kada korisnik vrši rezervaciju vozila, nakon rezervacije njemu je potrebna samo informacija da li ima slobodnih vozila, a pozadina toga ga ne zanima npr. lociranje slobodnih vozila, lociranje korisnika, sortiranje u najbolji poredak ko kriteriju koji je definisan pomoću Adapter paterna.

3.Dekorater patern

Ovaj patern koristimo prilikom dinamičkog dodavanja novih elemenata i ponašanja postojećim objektima. Omogućavaju se različite nadogradnje objektima koji svi u osnovi predstavljaju jednu vrstu objekta, bolje reći imaju osnovu. Umjesto da se definiše veliki broj izvedenih klasa, dovoljno je omogućiti različito dekoriranje objekata (tj. dodavanje različitih detalja), te se na taj način pojednostavljuje i rukovanje objektima klijentima, i samo implementiranje modela objekata.

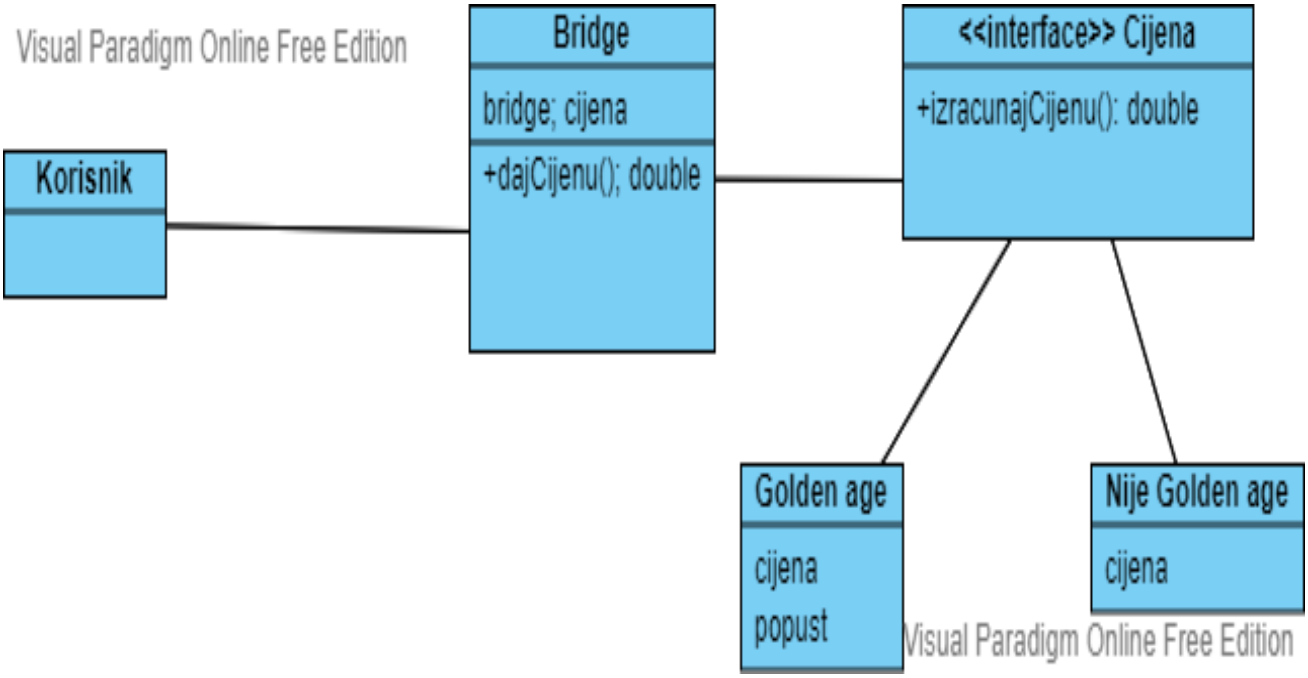
Ovaj patern bi se mogao iskoristiti ako bi htjeli da damo mogućnost korisniku da uredi svoj profil, npr. stavi opis profila ili svoju sliku uredi(npr.doda neki emoji ili okvir).

4.Bridge patern

Bridge patern se koristi da bi se omogućilo odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije. Bridge patern pogodan je kada se implementira nova verzija softvera a postojeća mora ostati u funkciji.

U naš sistem bi mogli ovo implementirati na sljedeći način:

- Klasa korisnik sadržo atribut DaLiJeGoldenage, u zavisnosti od vrijednosti ovog atributa, imamo obračunavanje popusta na različite načine za korisnika koji je Golden age I one koji nisu.
- Vraća se dva puta cijena samo se u slučaju dodatnih pogodnosti prikazuje I obračunati popust.



5.Composite patern

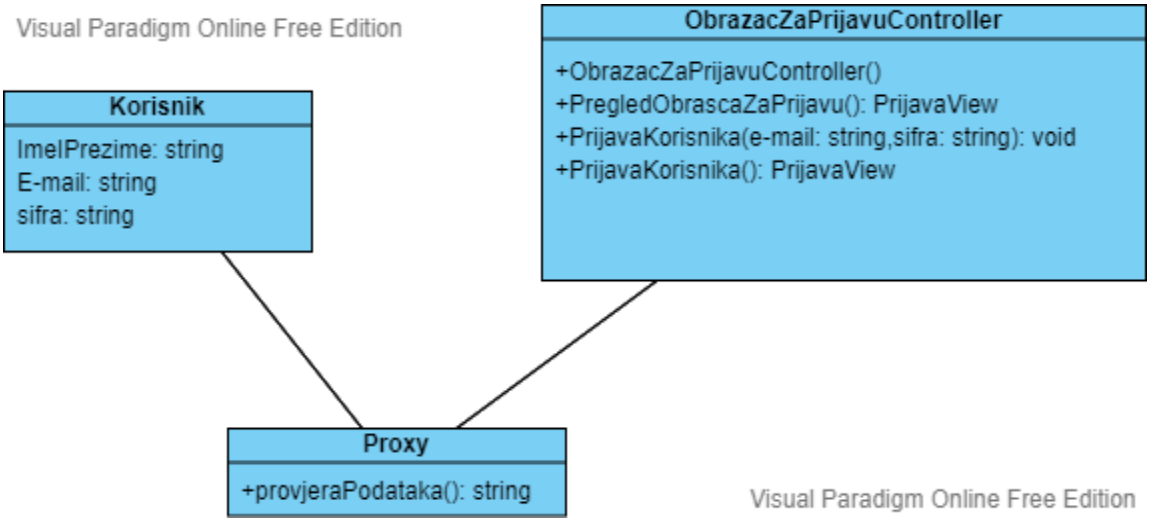
Ovaj patern služi da bismo postigli hijerarhiju objekata, tako što ćemo kreirati strukturu stabla pomoću klasa, u kojoj se kompozicije individualnih objekata (korijeni stabla) i individualni objekti (listovi stabla) ravnopravno tretiraju, odnosno moguće je pozvati zajedničku metodu nad svim klasama.

U našem sistemu ovaj patern možemo koristiti da se korisniku prikaže cijena njegove rezervacije, cijena se računa drugačije u zavisnoti od vrste vozila, te udaljenosti koja će se preći.

Ako ne bi koristili ovaj patern morali bi napraviti posebne klase za različite vrste vozila, koje bi onda na razlicite nacine implementirale metodu za izracunavanje cijene.

6.Proxy patern

Svrha Proxy patern-a je da omogući pristup i kontrolu pristupa stvarnim objektima. Proxy je obično mali javni surogat objekat koji predstavlja kompleksni objekat čija aktivizacija se postiže na osnovu postavljenih pravila. U našem sistemu možemo iskoristiti Proxy patern tako što bi ograničili prava pristupa određenim funkcionalnostima sistema. Dakle, nakon logina (ukoliko je korisnik uopšte registrovan) vršila bi se provjera podataka i na taj način bi se odredili prava pristupa nekim funkcionalnostima. Nakon logina bi se ustanovilo da li je korisnik Golden age ili ne, da li je registrovan ili je na stranici kao gost. Naravno, Golden age korisnik ima mnogo više funkcionalnosti od ostalih. Kako bi ovo funkcionisalo, dodat ćemo interfejs 'IproxyPrijava“ i klasu ProxyPrijava koja ima metodu preko koje se vrši provjera podataka i identifikacija korisnika na osnovu unesenih podataka.



7.Flyweight patern

Ovo je strukturalni patern koji bi mogli iskoristiti kada bi naši modeli imale neku osobinu koja se naknadno postavlja, te se toj osobini dodjeljuje neka default vrijednost koja se nalazi na samom sistemu radi smanjenog koristenja memorije. Spomenuta osobina nije značajan factor za samo odvijanje aplikacije. Trenutno dizajnirani sistem koristi samo podatke koji su zaista potrebni aplikaciji, te iz tog razloga ne vidimo neku potrebu za ovim paternom. Međutim, mogli bi ovaj patern iskoristiti za default sliku prilikom kreiranja novog korisničkog računa ili default sliku prilikom uvođenja novog vozila u naš sistem od strane administratora.