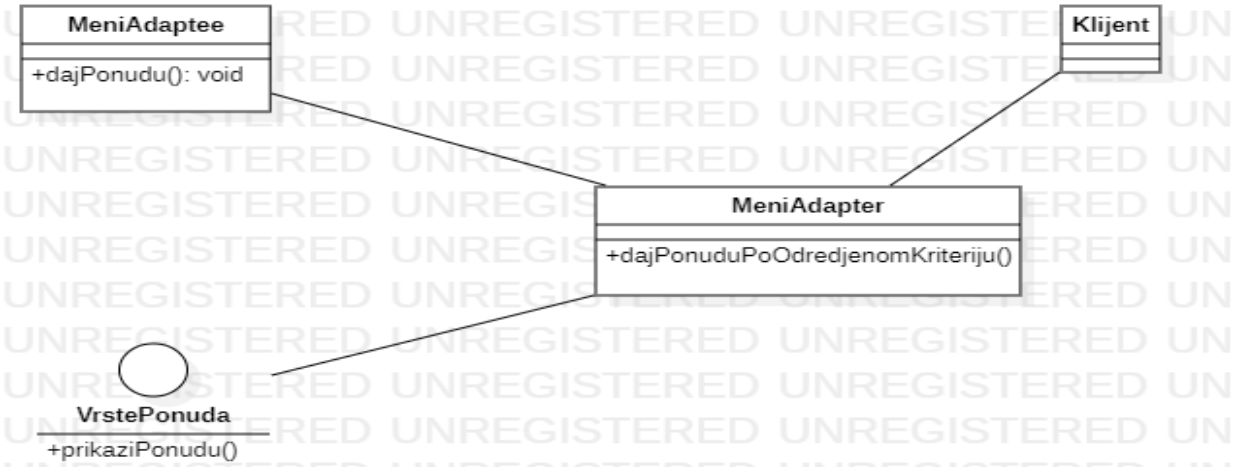
STRUKTURALNI PATERNI

1. Adapter

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa.

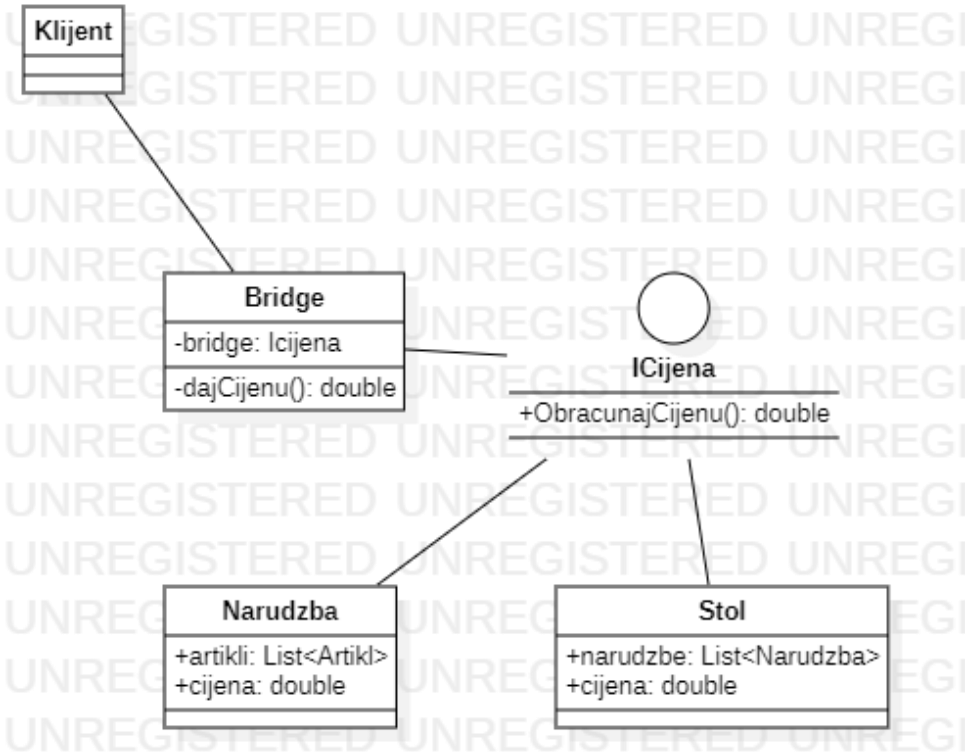
Nakon dodavanja atributa vrstaArtikla u našu postojeću klasu Artikl (kako bismo mogli dodati strukturne paterne) nadogradjujemo naš sistem kako bismo njegovom korisniku omogućili prikaz ponude po odredjenim kriterijima. Ustvari, omogućujemo korisniku da mu se ponuda prikaze samo za odredjene vrste artikala. Dakle, treba nadograditi našu već postojeću klasu Meni koja će samim time biti Adaptee klasa. Definisat ćemo novi interfejs VrstePonuda sa metodama koji pored vraćanja same ponude omogućuje vraćanje ponude samo po odredjenim vrstama artikla. Pored toga, definišemo i adapter klasu MeniAdapter koja će implementirati interfejs VrstePonuda. U metodama interfejsa će se pozivati metoda dajPonudu() klase Meni te će se zatim iz date liste artikala odvajati oni koji odgovaraju datoj vrsti te metode.



2. Bridge

Osnovna namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase tako da ta klasa može posjedovati više različitih apstrakcija i više različitih implementacija za pojedine apstrakcije.

Prvo dodajemo atribut cijena i metodu obracunajCijenu() u klasu Stol kako bismo mogli dodati Bridge patern. Klase Stol i klasa Narudzba obje imaju sličnu metodu koje računaju cijenu same narudžbe odnosno trošak stola nakon svih narudžbi koje su vezane za njega. Dodajemo interfejs ICijena koji računa cijenu, te Bridge klasu koja vraća cijenu u zavisnosti od toga da li je u pitanju sama narudžba ili sve narudžbe jednog stola.



3. Proxy

Proxy pattern bi mogao vršiti pravo pristupa samog uposlenika. Tačnije, ako uposlenik jednog lokala pokušava da pristupi informacijama drugog lokala (u kojem nije zaposlen) proxy će to da mu onemogući.

4. Decorator

Decorator patern bi se mogao implementirati da menadžer prilikom uploada slike lokala iste može editovat. U jednoj klasi bi se pronalazile metode za rezanje, rotiranje, promjenu veličine fotografije, dok bi se u drugoj klasi pronalazile metode za promjenu kontrasta, podešavanje osvjetljenja, oštrine itd.

5. Facade

Facde patern koristimo kako bi korisnicima pojednostavili korištenje složenog sistema koji se sastoji od više podsistema.

Za implementaciju našeg Facade paterna bismo koristili klasu Narudzba jer ona ima najviše komplikovanih aktivnosti koje obavlja a korisnik vidi samo krajnji rezultat, kao što je dodavanje Artikla, te bismo tu mogli napraviti klasu koja bi sve zajedno obuhvatila.

6. Flyweight

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

U našem sistemu primjena Flyweight paterna bi se mogla iskoristiti u slučaju kreiranja samog profila korisnika, menadžera ili uposlenika, kao i za kreiranje profila samog lokala. Odnosno, pošto se profilna slika ili slika kafića postavlja naknadno, radi smanjenjenja korištenja memorije uveli bismo default sliku za oba slučaja.

7. Composite

Composite patern se koristi da omogući formiranje strukture stabla pomoću klasa u kojoj se individualni objekti i kompozicije individualnih objekata jednako tretiraju.

Pošto se pojedinačne cijene (cijena za svaku narudžbu odvojeno) i konačni cijena (ukupna cijena koji se prikazuje mušteriji za stol nakon svih narudzbi) obje mogu prikazati mušteriji u ovom dijelu bismo mogli iskoristiti Composite patern. Naime, naša Component klasa bi sadržava cijenu same narudžbe ali Composite klasa bi imala listu cijena svih narudžbi za određeni stol.