Strukturalni dizajn paterni

Fasadni patern

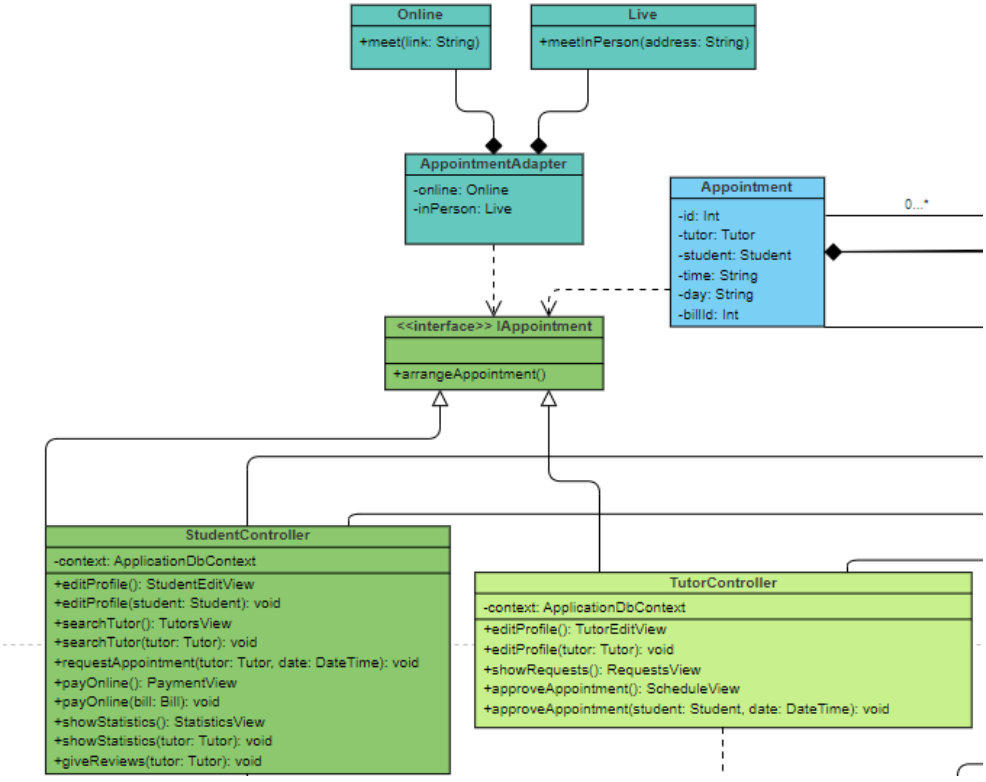
Fasadni patern koristimo kada želimo strukturirati podsistem u slojeve, gdje su apstrakcije i implementacije podsistema usko povezane. Postiže se prisustvo više interfejsa visokog nivoa na podsisteme čija implementacija nije vidljiva korisniku.

Ovaj patern možemo iskoristiti u slučaju srednje ocjene recenzije tutora, jer ukupna recenzija treba da bude dostupna svim korisnicima, dok implementacija zavisi od metode npr. getCurrentReview() koja uzima ocjene svih registrovanih korisnika.

Adapter patern

Adapter patern se koristi kada je potreban drugačiji interfejs već postojeće klase, a ne želimo je mijenjati, čime se omogućava njena šira upotreba.

Ovaj patern možemo iskoristiti za zakazivanje termina instrukcija. Naš sistem možemo dodatno unaprijedititi , tako što bi student osim mogućnosti da odabere termin instrukcija omogućili da unese i način pristupa instrukcija (uživo ili online) ukoliko tutor nudi obje opcije. Time bismo iskoristili adapter patern proširujući funkcionalnosti već postojeće klase.



Dekorater patern

Namjena dekorater paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja postojećim objektima. Ovaj patern možemo iskoristiti da unaprijedimo funkcionalnost pretrage tutora po kategorijama.

Možemo dodati nove kriterije sortiranja ili prilagoditi postojeće bez mijenjanja osnovne funkcionalnosti pretrage tutora. Također, dekoratori se mogu kombinovati kako bismo pružili još složenije mogućnosti sortiranja, prilagođene potrebama studenata.

Bridge patern

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.

Možemo ga implementirati u svrhu različitih plaćanja instrukcija, uživo ili putem interneta, tako što bismo klase Appointment i Bill spojili preko abstraktne klase npr. PaymentOptions. Osnovni način, tj. način uživo bi ostao isti, dok bi se prilikom online plaćanja uračunao popust.

Kompozitni patern

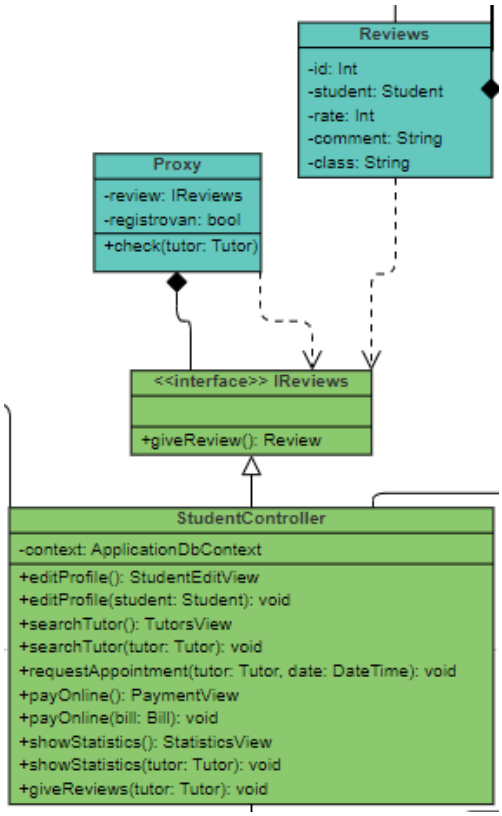
Osnovna namjena kompozitnog paterna je formiranje strukture drveta pomoću klasa, gdje se individualni objekti i kompozicije individualnih objekata jednako tretiraju.

Ovaj patern možemo koristiti u klasi Tutor kako bismo omogućili grupisanje tutora. Možemo ih grupisati po predmetima ili gradovima u kojim predaju, tako što tutore razvrstavamo po gradovima i onda tutore iz svakog grada dalje razvrstamo po predmetima koje predaju. Kompozitni objekat bi nam bio Tutori klasa, koja bi implementirala isti interfejs kao i pojedinačni aspekti. Klijent može pristupiti i razvrstanim tutorima, kao i kompozitnom objektu.

Proxy patern

Ovaj patern koristimo u situacijama kada je potrebno kontrolirati pristup objektu i obaviti dodatne provjere ili funkcionalnosti.

U ovom sistemu se može primijeniti na klasu Reviews kako bi se osiguralo da samo studenti koji su imali čas kod tog tutora mogu dodavati komentare i recenzije.



Flyweight patern

Flyweight patern treba da omogući da se svaki objekat na zahtjev obavi brzo. Ovaj patern koristimo kada imamo više objekata sličnih atributa i želimo optimizirati upotrebu memorije.

Možemo ga iskoristiti ukoliko korisnici ne unesu odmah profilnu sliku, da za sve studente u početku po defaultu bude jedna profilna slika, a za sve tutore druga.