STRUKTURALNI DIZAJN PATERNI

1. ADAPTER PATERN

Osnovna namjena Adapter paterna je da omogući širu upotrebu već postojećih klasa. Adapter patern se koristi kada je potreban drugačiji interfejs već postojeće klase, a ne želimo je mijenjati.

Naša aplikacija, odnosno naš sistem omogućava unos ocjene vožnje (Integer) od strane klijenta. Naš sistem možemo dodatno unaprijediti, tako što bi klijentu osim mogućnosti da unese ocjenu vožnje omogućili da unese i svoj komentar za tu vožnju. Time bi iskoristili adapter patern proširujući funkcionalnosti već postojeće klase.

2. FACADE PATERN

Osnovna namjena Facade paterna je da osigura više pogleda (interfejsa) visokog nivoa na podsisteme čija implementacija je skrivena od korisnika.

Ovaj patern smo upotrijebili i implementirali kod dodjeljivanja ocjene vozaču od strane klijenta. Nakon što klijent dodjeli ocjenu vozaču za odredjenu vožnju ta ocjena se mora evidentirati i mora se adekvatno dodijeliti odgovarajućem vozaču i takodjer mora biti dostupna na uvid budućim korisnicima našeg sistema.

3. DECORATOR PATERN

Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i ponašanja (funkcionalnosti) postojećim objektima.

Ovaj pattern smo upotrijebili na način kada se klijent registruje na naš sistem, on može uredjivati svoje korisničke podatke. Takodjer ovaj pattern je upotrijebljen i kod vozača koji takodjer mogu uredjivati svoje podatke, a isto tako i kod administatora koji ima pristup svim nalozima. Decorator patern je još upotrijebljen i kod firmi koje su registrovane u našem sistemu, jer one mogu zapošljavati i otpuštati svoje zaposlene.

4. 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. Bridge pattern nam omogućava da se iste operacije primjenjuju nad različitim podklasama.

Ovaj pattern smo upotrijebili prilikom kreiranja naloga na našem sistemu. Na naš sistem se možete prijaviti kao klijent ili kao vozač. Oba ova načina su ustvari iste operacije ali se primijenjuju na 2 različite klase, jedna je Klijent a druga Vozač.

5. PROXY PATERN

Osnovna namjena Proxy paterna je osiguranje pristupa resursima.

Ovaj patern smo upotrijebili prilikom prijave korisnika i/ili administartora na sistem. Prilikom prijave vrši se autentifikacija podataka i na taj način smo osigurali zaštićen pristup. Takodjer prilikom kreiranja novog računa od strane vozača iskoristili smo Proxy patern jer vršimo verifikaciju novog računa na osnovu vozačke dozvole.

6. COMPOSITE PATERN

Osnovna namjena Composite paterna (kompozitni patern) je da omogući formiranje strukture drveta pomoću klasa, u kojoj se individualni objekti (listovi stabla) i kompozicije individualnih objekata (korijeni stabla) jednako tretiraju.

Principi ovog patterna su zadovoljeni jer se naš sistem zasniva na vožnjama, koje se sastoje od više različitih aspekata. Recimo ovaj pattern je iskorišten jer se klijentu prikazuje procijenjena cijena vožnje na osnovu udaljenosti od željene destinacije i udaljenosti odabranog vozača od klijenta u trenutku narudžbe vožnje.

7. FLYWEIGHT PATERN

Flyweight patern treba da omogući da se svaki objekat na zahtjev može dobaviti brzo. Flyweight pattern omogućava programima da podrže ogromne količine objekata održavajući njihovu potrošnju memorije niskom. Uzorak to postiže dijeljenjem dijelova stanja objekta između više objekata. 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 korištenja memorije.

Ovaj patern možemo iskoristiti kod korisničkih profila u našem sistemu, tj. profila vezanih za vozača i klijenta. Recimo možemo postaviti default fotografiju korisničkog profila, tj. klijenta ili vozača, a da pritom oni mogu po želji dodati svoju fotografiju.