# Strukturalni paterni

**Facade** - Osnovna namjena Facade paterna je da osigura više pogleda visokog nivoa na podsisteme. Operacije koje su potrebne određenoj korisničkoj perspektivi mogu biti sastavljene od različitih dijelova podsistema. Pošto je naš sistem poprilično jednostavan, nema mnogo mjesta gdje bi mogli iskoristiti ovaj patern. Potencijalno bi ga mogli iskoristiti tako da sve naše funkcionalnosti smjestimo u jednu kontejnersku klasu sa listama objekata i metodama.

**Adapter** – njegova namjena se ogleda u tome da se već postojećim klasama poveća područje upotrebe, i to u slučajevima kada nam je potreban drugačiji interfejs a ne želimo mijenjati pomenutu klasu. To se postiže kreiranjem nove AdapterKlase koja ima ulogu posrednika između klase i interfejsa. Adapter patern bi se u našem slučaju mogao iskoristiti u situaciji kada nam treba lista svih slobodnih mjesta na parkingu, tako da adapter vrati listu sortiranu po određenom parametru, naprimjer udaljenosti. To bi postigli kreiranjem AdapterKlase koja bi preko interfejsa komunicirala sa ostatkom sistema.

**Decorator** – Osnovna namjena Decorator paterna je da omogući dinamičko dodavanje novih elemenata i funkcionalnosti postojećim objektima. U našem slučaju ovaj patern bismo mogli iskoristiti da omogućimo prikazivanje mape na više načina (reljef, saobraćaj, satelit). To bi postigli kreiranjem interfejsa 'IComponent' koji prepoznaje klasu koja treba biti dekorisana, i 'Decorator' klasu koja odgovara pomenutom interfejsu i implementira dinamički prošireni interfejs.

**Proxy** – Namjena Proxy paterna je da omogući pristup i kontrolu pristupa stvarnim objektima. U našem sistemu ovaj patern bi se mogao iskoristiti u svrhu provjere pristupnih podataka, ograničavanja prava pristupa određenim funkcionalnostima sistema u zavisnosti od toga da li je korisnik registrovan ili ne, da li je admin itd.
Da bismo to postigli, dodali bi interfejs 'IProxyLogin' i klasu ProxyLogin koja kroz metodu login vrši identifikaciju korisnika na osnovu korisničkog imena.

**Bridge** –Namjena Bridge paterna je da omogući primjenu istih operacija nad različitim podklasama. U našem projektu postoji mogućnost dobivanja raznih vrsta popusta u zavisnosti od uslova koje korisnik ispunjava, te se na osnovu toga cijena smanjuje za određeni iznos. Kako se svi popusti međusobno razlikuju mogli bismo napraviti interfejs 'iPopust' koji ima različite implementacije za svaki popust.

**Composite** – Osnovna namjena kompozitnog paterna je da omogući formiranje strukture stabla pomoću klasa, u kojoj se individualni objekti i kompozicije individualnih objekata jednako tretiraju. U našem projektu ovaj patern mozemo iskoristiti kod dobijanja cijene parkinga, koja zavisi od kategorije vozila za koje nam je potreban parking. U tu svrhu kreiran je interfejs 'IMjesto' koji definira interfejs-operacije za objekte u kompoziciji i implementira defaultno ponašanje koje je zajedničko za objekte oba tipa, te klase koja implementiraju interfejs korištenjem implementacija za pojedinačne komponente.

**Flyweight** – Koristi se u situacijama u kojima je potrebno da se omogući razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase tzv. glavno stanje, od dijela klase koji nije uvijek isti za sve određene objekte te klase tzv. sporedno stanje. U našem projektu ovaj patern možemo iskoristiti kod atributa slika, koji je automatski postavljen na defaultnu sliku, koja je zajednička za sve korisnike,ako nije naglašeno drugačije. Ovime se postiže racionalnija upotreba resursa, jer postoji samo jedna instanca defaultne slike u sistemu, koju koristi više korisnika.

**U naš sistem smo dodali Bridge i Composite pattern**.