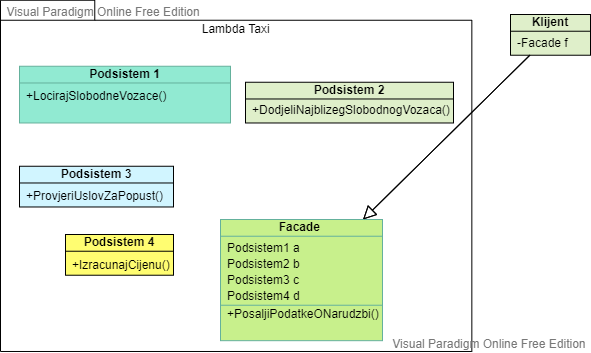
# STRUKTURALNI PATERNI

1. ***ADAPTER PATERN***

* Adapter patern služi da omogući da se interfejs već postojeće klase koristi kao drugi interfejs, ali često omogućava i to da postojeće klase rade sa drugim bez izmjene njihovog izvornog koda. On kreira novu klasu koja povezuje originalnu klasu i željeni interfejs.
* Ovaj patern možemo iskoristiti tako što ćemo napraviti jednu klasu Narudžba koja omogućuje i korisniku i ugostiteljskim objektima naručivanje taxi vozila iako se ne posmatraju na isti način. Korisnika gledamo kao jednu osobu, dok ugostiteljski objekti predstavljaju neku firmu, restoran, te tu narudžbu vrše u svoje ime za svog gosta.

1. ***FACADE PATERN***

* Facade patern služi kao prednji interfejs koji maskira složeniji kod tj. skriva složenost većeg sistema i klijentu pruža jednostavniji intefejs. Ovaj patern se koristi kada imamo neki kompleksan sistem koji ima mnogo međuzavisnih klasaili kojem je nepoznat izvorni kod.
* Ovaj patern možemo iskoristiti kod naručivanja vozila, jer korisnik ne zna na kojem principu mu sistem dodjeljuje vozača. Sistem prvenstveno locira sve slobodne vozače,zatim odredi koji je najbliži klijentu,te nakon toga provjerava da li korisnik ispunjava određene uvjete za popuste. Na osnovu udaljenosti između početne i krajnje lokacije odredi cijenu vožnje ,obračunatu sa popustom(ako postoji). Klijent na kraju dobije konačnu cijenu vožnje i podatke o vozaču ,iako ne zna koje sve operacije su bile potrebne za konačni rezultat.

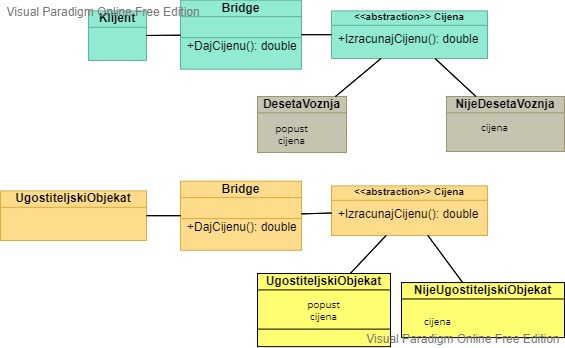


1. ***DECORATOR PATERN***

* Decorator patern služi da omogući da se dinamičko ponašanje dodaje za pojedinačne objekte i da isto ne utiče na ponašanje drugih objekata iz iste klase. Također omogućava i da se funkcionalnost podijeli između klasa u zavisnosti od područja interesa. Ovaj patern naslijeđuje originalnu klasu, ali se ne oslanja na nasljeđivanje prilikom dodavanja novih atributa i objekata, već kreira nove.
* Ovaj patern možemo iskoristi ukoliko se odlučimo da korisniku damo mogućnost promjene njegovih podataka, kao što je npr. promjena email-a ili kontakt telefona.

1. ***BRIDGE PATERN***

* Bridge patern služi da odvoji apstrakciju od implementacije na način da se one razlikuju i da budu nezavisne. Ima mogućnost korištenja naslijeđivanja kako bi razdvojio odgovornosti u različite klase. Ovaj patern je koristan kada imamo razliku između klase i onoga što ta klasa radi i u tom slučaju klasu možemo posmatrati kao apstrakciju, a ono što ona radi kao implemetaciju.
* Ovaj patern možemo iskoristi prilikom obračunavanja popusta na cijenu vožnje, zbog toga što korisnik i ugostiteljski objekat imaju različite popuste. Korisniku je svaka deseta vožnja besplatna, a ugostiteljki objekat prilikom naručivanja taxi vozila za svoje goste svaki put ima popust od 5%.



1. ***COMPOSITE PATERN***

* Composite patern služi da opisuje neku grupu objekata koji su tretirani na isti način i dio su klase, te da omogući kreiranje strukture stabla pomoću klasa. Ovaj patern još omogućava klijentima da pojedinačne objekte i njihove kompozicije tretira ujednačeno odnosno na isti način.
* Ovaj patern koristimo za izračunavanje cijene vozila, jer cijena vozila nikad nije ista, već se računa u zavisnosti od udaljenosti između početne i krajnje lokacije. Da ne postoji ovaj patern, trebala bi klasa za svaku udaljenost te njena fiksna cijena.

1. ***PROXY PATERN***

* Proxy patern služi da omogući kontrolu pristupa stvarnim objektima. Pored toga on se koristi i kao omotač kojeg klijent uvijek može pozvati da bi pistupio stvarnom objektu koji se nalazi „iza scene“, kao i kada klase imaju osjeltjive podatke ili spore operacije.
* Ovaj patern možemo iskoristi npr. kod prijave neke osobe u sistem. Prilikom unosa podataka isti ti podaci se provjeravaju, da li su ispravno uneseni i određuje se vrsta aktera našeg sistema. U zavisnosti od vrste aktera postoje različite funkcionalnosti koje su im namijenjene. Isto tako, klijent je jedini akter koji se ne može prijaviti u sistem bez prethodne registracije.

1. ***FLYWEIGHT PATERN***

* Flyweight patern služi kako bi različiti objekti imali isto glavno stanje, a svaki objekat ima drugačije sporedno stanje, te na taj način postižemo manje zauzimanje memorije. Ovaj patern još omogućava i lakšu implementaciju, testiranje, promjenu i ponovnu upotrebu.
* Ovaj patern možemo primjeniti kod statusa vozača, koji je pri samom kreiranju njegovog računa po default-u na „SLOBODAN“.