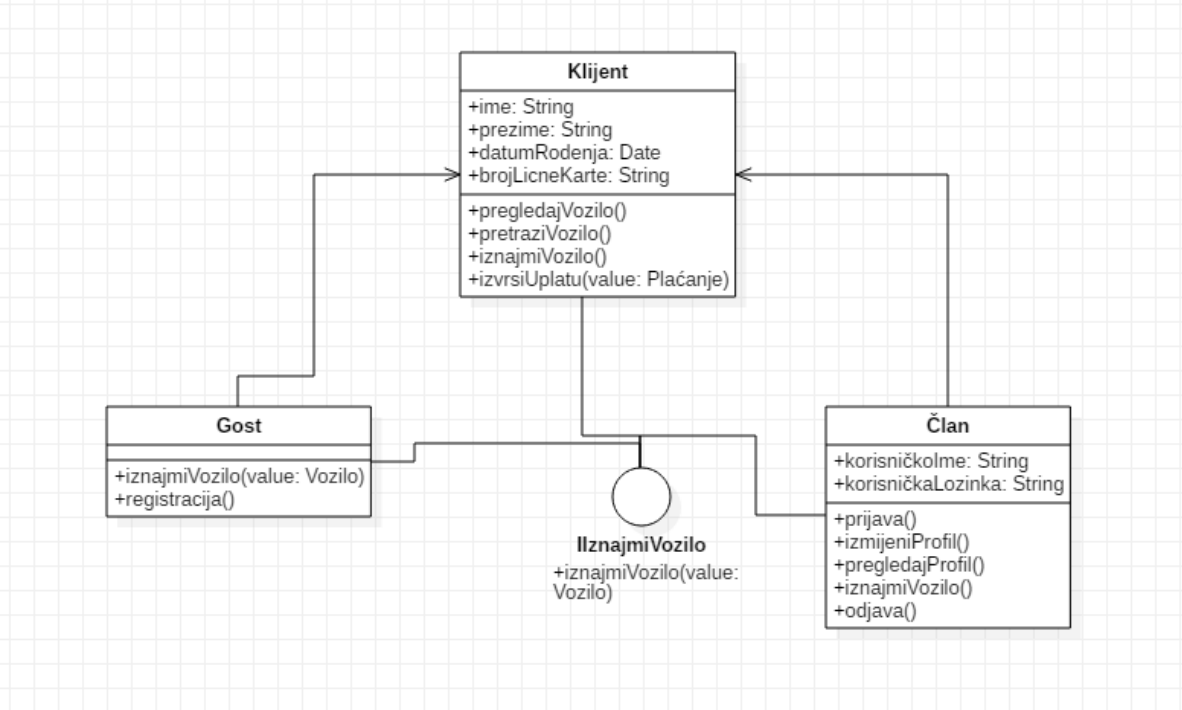
Strukturalni paterni

Paterne koji su iskoristeni u našem sistemu:

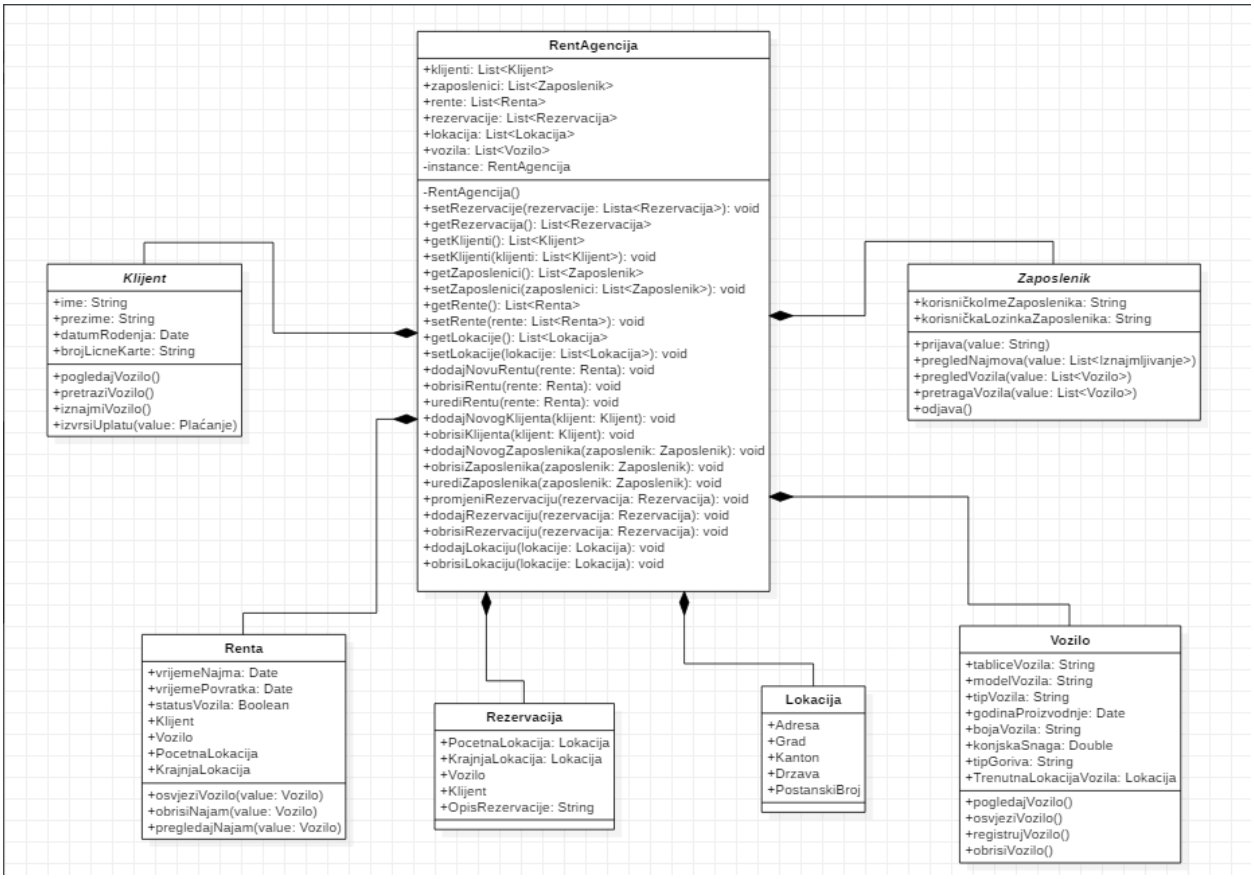
1. Composite patern

Composite patern služi za kreiranje hijerarhije objekata. Koristi se kada svi objekti imaju različite implementacije nekih metoda, no potrebno im je svima pristupati na isti način, te se na taj način pojednostavljuje njihova implementacija. U našem projektu imamo situaciju kada Gost i Član iznajmljuju vozilo.



2. Facade patern

Fasadni patern služi kako bi se klijentima pojednostavilo korištenje kompleksnih sistema. Klijenti vide samo fasadu, odnosno krajnji izgled objekta, dok je njegova unutrašnja struktura skrivena. Na ovaj način smanjuje se mogućnost pojavljivanja grešaka jer klijenti ne moraju dobro poznavati sistem kako bi ga mogli koristiti. Fasadni paterni je već primjenjen u našem sistemu. Korisnici aplikacije pozivaju metode koje žele da se izvrše, ali ne znaju koliko je kompleksna operacija i implementacija koja se krije iza poziva metode.



Paterni koji nisu iskorišteni u našem sistemu:

1. Flyweight patern: Flyweight patern koristi se kako bi se onemogućilo bespotrebno stvaranje velikog broja instanci objekata koji svi u suštini predstavljaju jedan objekat. Samo ukoliko postoji potreba za kreiranjem specifičnog objekta sa jedinstvenim karakteristikama (tzv. specifično stanje), vrši se njegova instantacija, dok se u svim ostalim slučajevima koristi postojeća opća instanca objekta (tzv. bezlično stanje). U našem sistemu nije moguće implementirati ovaj patern, ako pogledamo aktere Sistema svaki od njih ima svoju jedinstvenu osobinu.
2. Proxy patern: Proxy patern služi za dodatno osiguravanje objekata od pogrešne ili zlonamjerne upotrebe. Primjenom ovog paterna omogućava se kontrola pristupa objektima, te se onemogućava manipulacija objektima ukoliko neki uslov nije ispunjen, odnosno ukoliko korisnik nema prava pristupa traženom objektu. U našem sistemu se ne vrši nikakva manipulacija nad objetima kao ni kontorla pristupa. Nije bilo potrebe za uvođenjem ovog paterna. Da smo koristili kontrolu pristupa kod baze podata ili slično, tada bi mogli implementirati proxy patern.
3. Bridge patern: Bridge patern služi kako bi se apstrakcija nekog objekta odvojila od njegove implementacije. Ovaj patern veoma je važan jer omogućava ispunjavanje Open-Closed SOLID principa, odnosno uz poštivanje ovog paterna omogućava se nadogradnja modela klasa u budućnosti te osigurava da se neće morati vršiti određene promjene u postojećim klasama. U našem sistemu, Bridge patern mogli bi iskorisitit za slučaj da u sistemu imamo vise vrsta uposlenika i da čuvamo podatke o plati tih uposlenika te za slučaj da računamo platu pri čemu bi zaposlenici imali zajedničke podatke. Pošto ne čuvamo podatke o plati niti imamo vise vrsta uposlenika nema potrebe za ovođenjem ovog paterna.