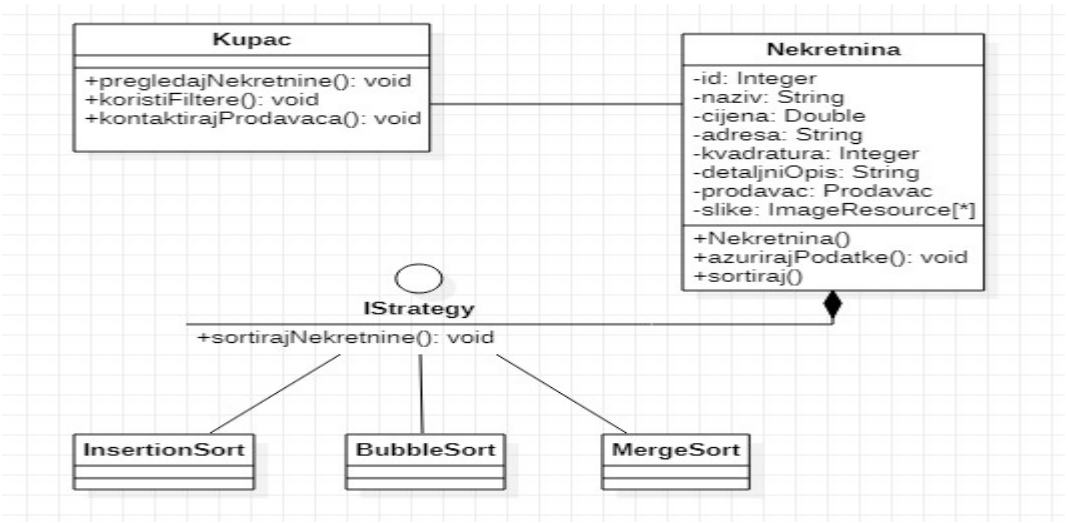
# PATERNI PONAŠANJA

## Strategy patern

Strategy patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase. Pogodan je kada postoje različiti primjenjivi algoritmi (strategije) za neki problem. Strategy patern omogućava klijentu izbor jednog od algoritma iz familije algoritama za korištenje.

Ovaj patern bi mogli iskoristiti pri sortiranju nekretnina da bi se uvjerili koji će algoritam najbrže sortirati. To bi mogli napraviti na način da dodamo u klasu Nekretnina metodu sortiraj(). Nakon toga bi napravili novi interface IStrategy. Tada bi mogli bez većih problema dodati nove klase koje će naslijediti IStrategy interface bez mijenjanja postojećih klasa. Iz IStrategy interface-a bi imali mogućnost svih vrsti sortiranja.




## State patern

State patern je dinamička verzija Strategy paterna. Objekat mijenja način ponašanja na osnovu trenutnog stanja. Postiže se promjenom podklase unutar hijerarhije klasa.

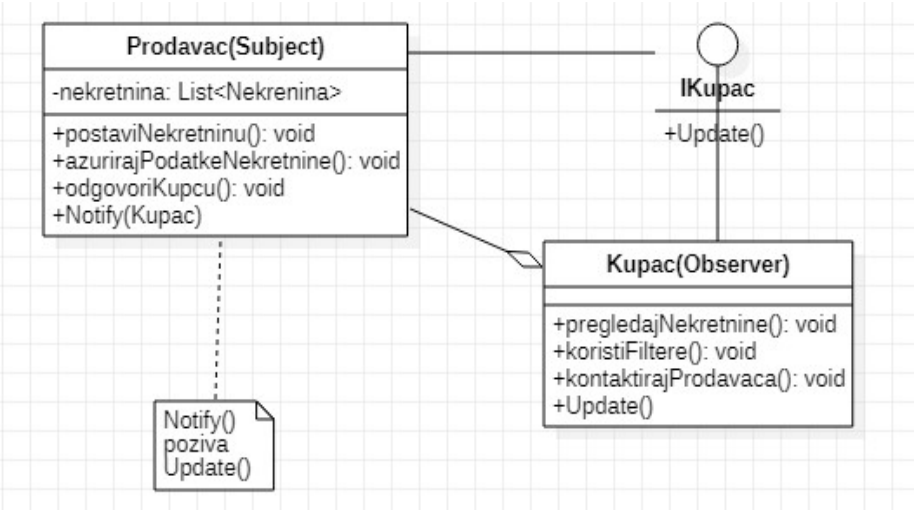
Ovaj patern u sistemu već imamo. Koristimo ga za način prikazivanja nekretnina tj. ako je korisnik prijavljen na sistem moći će kupiti nekretninu, dok neprijavljeni korisnik to neće moći uraditi. Implementiran je interfejs IState koji će prikazivati pogled da li je

korisnik prijavljen ili ne. Potrebni su pogledi PogledRegistrovanog() i PogledNeregistrovanog() za realizaciju u sistemu.

## Observer patern

Uloga Observer paterna je da uspostavi relaciju između objekata tako kada jedan objekat promijeni stanje drugi zainteresirani objekti se obavještavaju.

Ovaj patern bi mogli iskoristiti na način ako korisnik željenu nekretninu stavi u "Favorite" i pri tome se desi promjena cijene ili prodavac označi da je ta nekretnina na popustu, tada bi došla obavijest Kupcu za određenu promjenu. Subject bi u ovom slučaju bio prodavac koji bi mijenjao cijenu nekretnine, Observer bi bio korisnik koji je stavio nekretninu u "Favorite". Update bi predstavljao prijem obavještenja npr. preko e-maila. Notify bi bio vrsta e-mail poruke koji bi korisnici dobili kada označe nekretninu, a State bi bilo novo stanje.




## Iterator patern

Iterator patern omogućava sekvencijalni pristup elementima kolekcije bez poznavanja kako je kolekcija struktuirana.

U našem sistemu se može primijeniti pri upotrebi filtera prilikom pretraživanja nekretnina tj. u sistemu već postoje filteri za pretragu koji bi ukoliko korisnik želi prikazao nekretnine po vrsti tj. po Stanu, Kući, Vikendici itd.

## Medijator patern

Medijator patern enkapsulira protokol za komunikaciju među objektima dozvoljavajući da objekti komuniciraju bez međusobnog poznavanja interne strukture objekta.

Korisnici da bi stupili u kontakt mogu koristiti neki vid chat platforme. U našem slučaju bi mogli komunicirati samo registrovani korisnici. Ukoliko poruke imaju neprimjeren sadržaj, ili neprimjerene riječi će se odbacivati.

## Chain of responsibility patern

Chain of responsibility patern namijenjen je kako bi se jedan kompleksnii proces obrade razdvojio na način da više objekata na različite načine procesiraju primljene podatke.

Ovaj patern bi mogli iskoristit na način kada bi dodali rezervaciju nekretnine u aplikaciju. Ako se korisnik odluči da rezerviše nekretninu, potrebno je prvo da pošalje zahtjev za rezervaciju koju treba da odobri Admin sistema. Morali bi u sistem dodati novu klasu za zahtjev, klase koje će nasljeđivati tu novu klasu i interface IHandler.

## Template method patern

Omogućava izdvajanje određenih koraka algoritma u odvojene podklase. Struktura algoritma se ne mijenja - mali dijelovi operacija se izdvajaju i ti se dijelovi mogu implementirati različito.

Ovaj patern možemo iskoristiti kod sortiranja nekretnine po cijeni i lokaciji, tako što ćemo algoritam sortiranja izdvojiti u posebnu klasu.

## Interpreter patern

Interpreter patern podržava interpretaciju instukcija napisanih za određenu upotrebu.

U našem sistemu ovaj patern možemo iskoristiti za provjeru validnosti šifre pri registraciji korisnika (npr. da li je uneseno dovoljno znakova, da li su velika i mala slova, da li ima brojeva i drugih znakova).