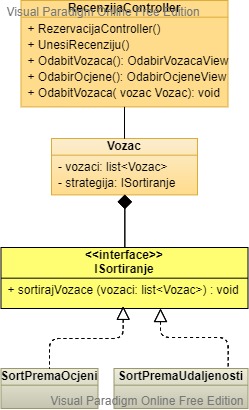
# PATERNI PONAŠANJA

1. *STRATEGY PATERN*

* Strategy patern je obrazac ponašanja softvera dizajna koji omogućava odabir algoritma u vremenu izvođenja. Kod prima instrukcije o tome koji od algoritama u porodici koristiti, umjesto direktnog implementiranja jednog algortima. Ovaj patern izdvaja algoritam iz matične klase i uključuje ga u posebne klase.
* U našem sistemu ovaj patern možemo primjeniti npr. za sortiranje vozača po ocjeni i po broju odrađenih sati.

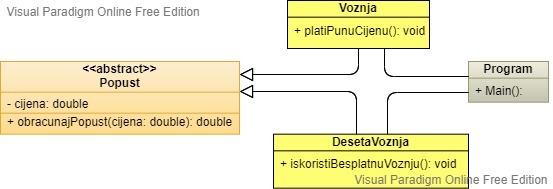
**

1. *STATE PATERN*

* State patern je obrazac ponašanja softverskog dizajna koji omogućava objektu da primijeni svoje ponašanje kada se njegovo unutrašnje stanje promijeni. Jako je blizak Strategy paternu i mijenja način ponašanja na osnovu trenutnog stanja.
* U našem sistemu ovaj patern možemo primjeniti npr. kod statusa vozača. Ukoliko status vozača nije „DOSTUPAN“ taj vozač neće sigurno biti izabran i poslan nekom korisniku koji je naručio vožnju. Čim korisnik završi svoju vožnju i unese to u sistem, njegov status sa „ZAUZSET“ prelazi na „SLOBODAN“ i sistem ga uzima u razmatranje.

1. *TEMPLATE METHOD PATERN*

* Template method patern je obrazac ponašanja softverskog dizajna i nalazi se u superklasi, obično apstraktnoj, te definiše kostur operacije kao slijed brojnih koraka visokog nivoa. Ovi koraci su sami implementirani dodatnim metodama u istoj toj klasi.
* U našem sistemu ovaj patern možemo primjeniti npr. kod obračunavanja popusta prilikom naručivanja taxi vozila. Proces naručivanja se odvija za sve korisnike isto, osim za podklasu korisnika kojima je naručena vožnja deseta vožnja.



1. *OBSERVER PATERN*

* Observer patern je obrazac ponašanja softverskog dizajna u kojem objekat nazvan subjektom, održava listu svojih posmatrača i obavještava ih o svim promjenama stanja.
* U našem sistemu ovaj patern možemo primjeniti npr. kod slanja obavještenje korisniku ukoliko dođe do nekih nepredviđenih okolnosti i vozač bude spriječen doći do njega tj. obavještavamo ga o promjeni stanja.

1. *ITERATOR PATERN*

* Iterator patern je obrazac ponašanja softverskog dizajna u kojem se iterator koristi za prolazak kroz kontejner i pomoću njega pristupamo elementima kontejnera.
* U našem sistemu ovaj patern možemo primjeniti npr. za filtriranje vozača po ocjeni 5 i prikazat će se svi vozači sa ocjenom 5.

1. *CHAIN OF RESPONSIBILTY PATERN*

* Chain of responsibilty patern je obrazac ponašanja softverskog dizajna koji vam omogućuje prosljeđivanje zahtijeva duž cijelog lanca rukovatelja. Nakon primanja tog zahtjeva, rukovatelj može obraditi taj zahtjev ili ga proslijediti dalje. Pomoću ovog paterna omogućavamo razdvanje procesa na način da se nezavisno jedan od drugog mijenjaju.
* U našem sistemu ovaj patern možemo primjeniti npr. kada korisnik želi naručiti taxi vozilo, To bi se uradilo na način da korisnik prvo mora unijeti sve neophodno podatke, te bi mu se potom ponudila cijena. Nakon toga korisniku bi bila ponuđena opcija za naručivanje tog vozila ili za otkazivanje vožnje.

1. *MEDIATOR PATERN*

* Mediator patern je obrazac ponašanja softverskog dizajna koji definiše objekat koji obuhvata način interakcije skupa više objekata odnosno daje mogućnost komuniciranja objekata.
* U našem sistemu ovaj patern možemo primjeniti npr. kada vozač komunicira sa adminom i šalje obavještenje o spriječenom dolasku po klijenta. Mediator je admin koji te poruke šalje korisnicima, jer bi nastao haos kada bi vozači i korisnici komunicirali, jer bi se moglo desiti da vozač pošalje pogrešnom korisniku obavještenje.