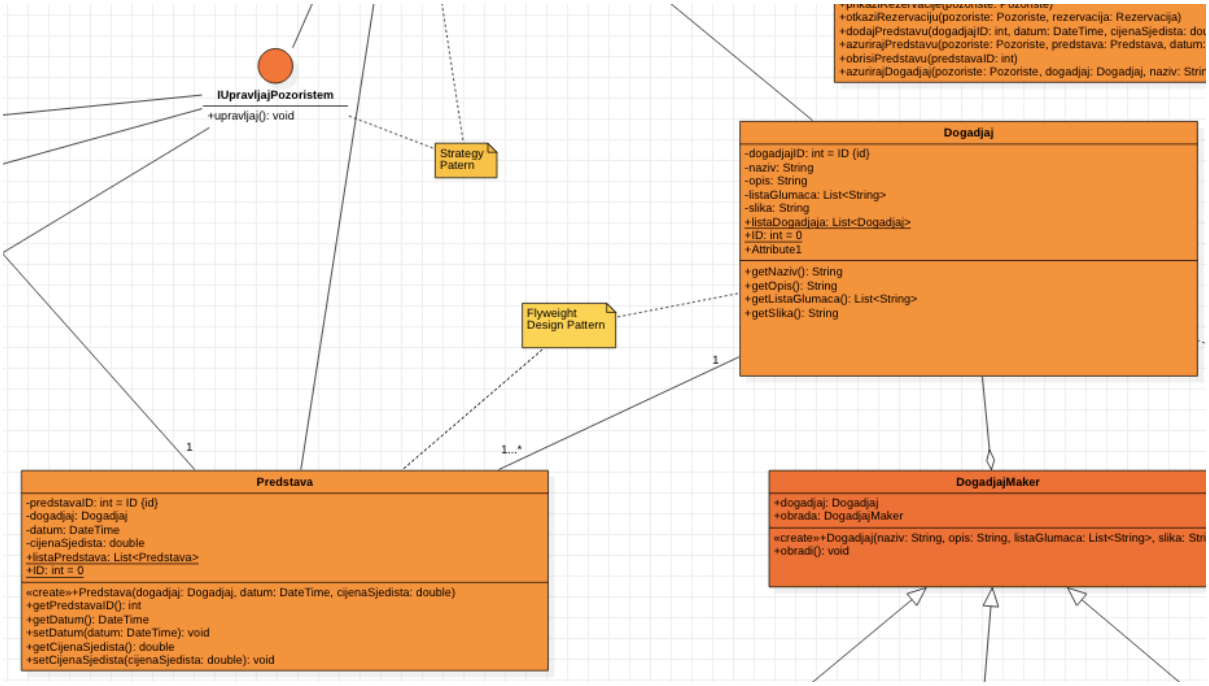
Strukturalni paterni

- Flyweight patern
  - Ovaj patern se koristi kada je potrebno omogućiti razlikovanje dijela klase koji je uvijek isti za sve određene objekte te klase. S obzirom na činjenicu da može da se desi izvedba iste predstave (sa istim nazivom, listom glumaca i sl) ali sa različitim datumom i vremenom izvođenja, odlučile smo odvojiti početnu klasu Predstava u dvije klase – Predstava i Dogadjaj. Glavno stanje (razlikuje se za različite objekte) je predstavljeno klasom Predstava, dok je sporedno stanje (isto za određene objekte) predstavljeno klasom Dogadjaj. Ovaj patern se pokazao u ovom slučaju veoma korisnim. Na primjer, ukoliko admin iz nekog razloga odluči zamijeniti nekog glumca iz liste glumaca klase Predstava, takav posao bi bio mnogo duzi bez klase Dogadjaj. Admin bi bio primoran u svim instancama klase Predstava (koje se razlikuju samo po vremenu) izvršiti izmjene. Međutim, sa klasom Dogadjaj, takva izmjena bi se izvršila samo jednom jer svaka instanca klase Predstava ima instancu na Dogadjaj kao svoj atribut.



- Bridge patern
  - Namjena Bridge paterna je da omogući odvajanje apstrakcije i implementacije neke klase. Klijent klasa Rezervacija treba da dobije ukupni trošak (njen atribut - ukupniTrosak) za sva sjedišta rezervisana od strane jednog korisnika za jednu predstavu. Taj trošak je različit ukoliko je korisnik Kupac (nema popusta) ili PremiumKupac (ima popusta koji zavisi od tipa njegove kartice). Obje te klase će imati različitu implementaciju metode izračunajPopust. To je posao koji je bolje da za nas dobavi apstrakcija - Bridge klasa. Ona će na osnovu atributa tipa ITrosak (to ce biti Kupac ili PremiumKupac), atributa cijenaSjedista i ukupniBrojSjedista dobiti odgovarajuću vrijednost ukupnog troška.