Solid principi

1. *Princip pojedinačne odgovornosti*

Ovaj princip je zadovoljen u vecini klasa jer svaka klasa ima pojedinačnu odgovornost. Svaka klasa je osposobljena za jednu i iskljucivo jednu funkciju. Jedina klasa koja bi mogla da narušava ovaj princip je klasa **SistemBazePodataka.** Međutim, to nije nikakav problem. Naime, prilikom implementacije ova klasa predstavlja Bazu podataka. Sve metode vezane za ovu klasu će biti realizirane kao **upiti** nad tom bazom podataka.

1. *Otvoreno zatvoreni princip*

Ovaj princip je ispoštovan jer sve klase koje za atribute imaju druge klase, ne vrše modifikaciju istih već samo koriste njihove metode ili ih koriste kao atribute u nekoj listi. Na mjestima gdje bi ovaj princip mogao biti narušen uvedeni su interfejsi koji razdvajaju klasu od baze. Jedan od primjera bi bio obračunavanje popusta za kupca koji je otvoren za nadogradnju (način obračnavanja popusta), ali zatvoren za promjene (promjene klase kupac).

1. *Liskov princip zamjene*

Liskov princip zamjene omogućava implementaciju polimorfizma. U našem slučaju ovaj princip je zadovoljen kreiranjem bazne klase Korisnik koju nasljeđuju klase **Vozac, Admin** i **Kupac**. Tamo gdje se sve ove klase mogu nalaziti i koristiti, moguće je koristiti i baznu klasu Korisnik. Ove klase se isto ponašaju pri pregledu sadržaja (vožnji).

1. *Prinicip izolacije interfejsa*

Ovaj princip je zadovoljen pomoću specificiranih interfejsa. Imamo posebne interfejse za posebne funkcionalnosti i svaka vrsta korisnika implementira interfejs koji joj je potreban. Zbog toga interfejs koji se tiče vožnji je podijeljen na interfejs za pregled i interfejs za modifikaciju vožnju.

1. *Princip inverzije ovisnosti*

U našem slučaju bazna klasa Korisnik je apstraktna i iz nje su izvedene klase **Vozac, Admin** i **Kupac**. Time je ovaj princip zadovoljen.