SOLID Principi

1. **Single Responsibility Principle - SRP (Princip pojedinačne odgovornosti)**

   - KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU
   - Napravili smo klase koje obavljaju samo jednu funkciju. Klasa Account sadrži samo podatke vezane za profil (ime,prezime,korisnicko ime,password,email). Klasa vozilo sadrži samo podatke o vozilima (id,model,cijena,slika) I sl.
   - Na našem dijagramu klasa, svaka klasa ima pojedinačnu odgovornost, pa je shodno tome prvi princip ispoštovan

2. **Open Closed Principle – OCP (Otvoreno – zatvoreni princip)**

   - Mi u našem sistemu imamo dvije apstraktne klase Vozilo i Account koje nasljeđuju određene klase I nadogradnja tih klasa ne mijenja apstraktnu logiku klasa Vozilo I Account
   - Princip je ispoštovan, s obzirom da se nadogradnja modula ne bi odrazila na vezu između klasa I entiteti softvera su otvoreni za nadogradnju, ali zatvoreni za modifikacije

3. **Liskov Substitution Principle – LSP (Liskov princip zamjene)**

   - Po LSP principu podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima
   - Naš projekat ne narušava Liskov princip zamjene jer smo ispoštovali date korake
   - Npr: da smo stavili +setPlata() ili +setDatumZaposlenja() u klasu Account onda bi sve naslijeđene klase morale da implementiraju tu metodu, a npr. znamo da Korisnik ne prima nikakvu plaću niti mu treba Datum zaposlenja pa bi onda jedino rješenje bilo da mu setamo to automatski na “0KM” ili neki datum I tad bi bio narušen princip

4. **Interface Segregation Principle – ISP (Princip izoliranja interfejsa)**

   - Izvedene klase koriste sve metode sadržane u baznim klasama, stoga ne postoje suvišne metode koje korisnik neće upotrebljavati
   - Nismo imali potrebe za pravljenjem interfejsa jer su ovo sve podatkovne klase, a radnje će se odvijati u kontrolerima
   - ISP princip nije narušen u našem dijagramu

5. Dependency Inversion Principle – DIP (Princip inverzije ovisnosti)

- Po DIP principu ne treba ovisiti od konkretnih klasa, te prilikom nasljedivanja treba razmotriti slučaj da je bazna klasa apstraktna.
- U našem projektu imamo klase Korisnik, Vozač, Administrator koje se nasljeđuju iz apstraktne klase Account.
- Na ovaj način smo postigli da system neće biti osjetljiv na promjene, tj. nas model visokog nivoa ne ovisi od modela niskog nivoa, već ovise od apstrakcije po čemu smo zadovoljili DIP princip