SOLID PRINCIPI

- **Single responsibility Principle** - princip pojedinačne odgovornosti, tvrdi da svaka klasa treba biti zadužena samo za jednu odgovornost unutar projekta.
  Svaka klasa je napravljena samo zbog jednog razloga, kako bi obavljala samo jedan zadatak, odnosno služila jednom cilju.
  Neki primjeri pojedinačnih odgovornosti:
  1. Klasa Administrator je zadužena za sve promjene koje se odnose na ponudu letova, što uključuje kreiranje, brisanje i uređivanje leta.
  2. Klasa Let je zadužena za čuvanje podataka o određenom letu (broj putnika, vrijeme polijetanja, vrijeme slijetanja …). Također klasa Let povezuje klase Karta, Avion i Destinacija.
- **Open/Closed Principle** - otvoreno zatvoren princip, klasa treba biti otvorena za nadogradnje, ali zatvorena za modifikacije
  Uzmimo za primjer apstraktnu klasu Korisnik. Iz te klase postoje tri izvedene klase - Putnik, Radnik i Administrator. Promjenom klasa Putnik i/ili Radnik i/ili Administrator tj. dodavanjem atributa ili metoda koje dodatno definišu te klase mi ne moramo mijenjati klasu Korisnik.
- **Liskov Substitution Principle** - Liskov princip zamjene, svi podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima bez da to utječe na ispravnost rada programa
  Iz klase Korisnik postoje tri izvedene klase - Putnik, Radnik i Administrator. Metode klase Korisnik su takve da imaju smisla da se pozivaju i nad klasama Putnik, Radnik ili Administrator jer predstavljaju samo kreiranje korisnika i gettere i settere.
- **Interface Segregation Principle** - princip izoliranja interfejsa, bolje je imati više specifičnih interfejsa, nego jedan generalizovani odnosno klijenti ne treba da ovise o metodama koje neće upotrebljavati. Ovo čuva korisnika od promjena metoda koje ga se ne tiču.
  S obzirom da interfejsi nisu korišteni, ovaj princip je ispoštovan.
- **Dependency Inversion Principle** - princip inverzije ovisnosti, sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama.
  Apstraktne klase i interfejsi mnogo se manje mijenjaju nego njihove konkretne izvedenice. Stoga je bolje ovisiti o apstrakcijama nego o stvarnim klasama. Slijedeći ovaj princip smanjuje se utjecaj koji promjena može imati na sistem.
  Ovaj princip je ispoštovan jer je bazna klasa Korisnik ujedno i apstraktna klasa iz koje su naslijeđene klase Korisnik, Radnik i Administrator.