1. **Princip pojedinačne odgovornosti :** KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU

Klase u našem sistemu imaju pojedinačnu odgovornost. Odvojili smo sve koncepte u njihove vlastite klase tako da svaka klasa ima jedan i samo jedan razlog za promjenu. Npr. klasa ClanUprave može da izbriše autobus, ali se ne bavi time kako si te obavlja u samoj bazi podataka.

1. **Otvoreno zatvoren princip :** ENTITETI SOFTVERA (KLASE, MODULI, FUNKCIJE) TREBALI BI BITI OTVORENI ZA NADOGRADNJU, ALI ZATVORENI ZA MODIFIKACIJE

Ovaj princip je primjenjen npr. kod klase ClanUprave gdje dati interfejs razdvaja klasu ClanUprave od API-ja baze i omogućava nam da mijenjamo okruženje baze bez bilo kakvog utjecaja na klasu ClanUprave.

1. **Liskov princip zamjene** : PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA

Apstraktna klasa Up*oslenik ,* odnosno njeni podtipovi su zamjenjivi u svakom trenutku, jer svi atributi aps. klase važe za sve klase koje su izvedene iz nje, odnosno sve izvedene klase su zaista uposlenici, te sve ono što je u apstraktnoj klasi važi i za izvedene klase.

1. **Princip inverzije ovisnosti** : **A.**MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA. OBA BI TREBALO DA OVISE OD APSTRAKCIJA. **B.** MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD APSTRAKCIJA.

Veliki broj klasa ovisi o interfejsima ( *DataBase )* i od apstraktne klase (Uposlenik) jer se ova klasa rijetko mijenja čime je zadovoljeno da klase visokog nivoa ne ovise o klasama nižeg nivoa, kao i ovisnost detalja od apstrakcija.

1. **Princip izoliranja interfejsa :** KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI.

U našem sistemu ne postoje klase, a niti interfejsi koji implementiraju veliki broj metoda, već samo one neophodne koje su potrebne u njihovom radu, te na taj način oni ne ovise od metoda koje ne upotrebljavaju. Korisnici su zaštićeni od metoda koje im ne trebaju, dajući im interfejse prema onim metodama koje su im potrebne.