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

Niti jedna klasa nije opterecena metodama za koje direktno nije vezana. Dakle, primjer klasa *Sastanak* brine samo o sastanku, odnosno glavnim atributima vezanim za sastanak, a ne vodi racuna recimo o pristupu bazi podataka.

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

Ovaj princip je posebno primjenjen kreiranjem klase *Specifikacija* gdje je omogucena nadogradjnja (dodavanje specifikacija za odredjeni artikal) a da pri tome ne dolazi do modifikacije klase *Artikal.*

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

Apstraktna klasa *Artikal,* odnosno njeni podtipovi su zamjenjivi u svakom trenutku. Klasom *Specifikacija* je to jos dodatno omoguceno, jer nije vazno koji podtip selektujemo dobit cemo njegovu specifikaciju.

1. **Princip inverzije ovisnosti** : **A.**MODULI VISOKOG NIVOA NE BI TREBALI O VISITI 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.

Vecina klasa ovisi o interfejsima ( *DataBase )* i apstraktnim klasama ( *Artikal, Osoba )* jer ove klase se rijetko mijenjaju cime je zadovoljeno da klase visokog nivoa ne ovise o klasama nizeg nivoa, kao i ovisnost detalja od apstrakcija.

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

Ne postoji interfejs ili apstraktna klasa sa velikim brojem metoda, vec su razdvojene po funkcionalnostima koje trebaju da izvrsavaju, tako izvedene klase ne moraju da implementiraju metode koje ih se ne ticu, te promjene tih metoda nece direktno utjecati na njih.