Analiza SOLID principa

* Princip pojedinacne odgovornosti

KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU

Sa dijagrama se vidi da je zadovoljen ovaj princip jer klase brinu o pojedinacnim stvarima npr. klasa Kanal brine samo o kanalim, Paket telefonije samo o fiksnoj telefoniji itd. Zbog toga su pogodne za modifikaciju.

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

Klase je moguće nadograditi odnosno modifikovati postojeće klase dodavanjem novih funkcionalnosti.

* Liskov princip zamjene

PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA

Iz apstraktne klase Osoba naslijeđuju se klase Korisnik i Uposlenik i dijele nekoliko zajedničkih atributa ali se bez problema moze umjesto klase Korisnik korisititi i objekat tipa Osoba sto potvrđuje da je ispoštovan ovaj princip.

* 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.

Ovaj princip je ispoštovan jer klase ne ovise o konkretnim klasama već ili o apstraktnoj klasi ili u nekim slučajevima o interfejsu.

* Princip izoliranja interfejsa

KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI.

Na našem dijagramu klasa ne postoji interfejs ili klasa sa velikim brojem metoda, već su sve raspoređene ovisno o funkcionalnosti koju izvršavaju. Na taj način samo se zaštitili da klase ne budu ovisne o metodama koje neće upotrebljavati.