**SOLID PRINCIPI**

**Princip pojedinacne odgovornosti :**

*KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU*

Klase nisu opterecene metodama niti objektima za koje nisu direktno vezane. Na primjer klasa Menadžer brine samo o poslovima koji se tiču menadžera tj. upošljavanju, ažuriranju turističkih ponuda i pregledu izvještaja.

**Otvoreno zatvoren princip :**

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

Ovaj princip je primjenjen kreiranjem na klasu Osobagdje je omogućena nadogradnja, s obzirom da imamo veliki broj različitih aktera u sistemu.

**Liskov princip zamjene** :

*PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA*

Apstraktna klasa Osoba*,* odnosno njeni podtipovi su zamjenjivi u svakom trenutku. Klijent može pisati recenziju, što može i neka osoba. Isto važi za apstraktnu klasu Izvještaj, s obzirom da imamo više različitih vrsta izvještaja koje su potrebni menadžeru i administraciji.

**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.*

Većina klasa ovisi o interfejsima, te apstraktnim klasama jer ove klase se rijetko mijenjaju čime je zadovoljeno da klase visokog nivoa ne ovise o klasama nižeg nivoa, kao i ovisnost detalja od apstrakcija. Npr. klase Osoba i Izvještaj.

**Princip izoliranja interfejsa :**

*KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI.*

Nijedan interfejs ili apstraktna klasa nema prevelik broj metoda, već su razdvojene po funkcionalnostima koje trebaju da izvrsavaju, i tiču se samo jednog objekta. Tako da izvedene klase ne moraju da implementiraju metode koje nemaju veze s njima pa ni promjene tih metoda neće direktno utjecati na njih.