## SOLID principi

Princip pojedinačne odgovornosti – Na dijagramu se može jasno primijetiti da je zadovoljen ovaj princip. Svaka klasa radi samo one funkcije za koje je zadužena. Niti jedan klasa nema metoda koje se brinu o stvarima za koje ona nije zadužena. Npr. klasa Zadaca ne zna ništa drugo osim onoga što se vezuje za zadaću kao što su podaci TipZadace, Bodovi, KrajnjiRok te manipulacija ovim podacima.

Otvoreno zatvoreni princip – Što se tiče MVC patterna ovaj princip je zadovoljen to možemo vidjeti kod dodavanja view-ova. Sam interfejs IView je zatvoren za modifikacije ali veoma otvoren za nadogradnju. Također npr. apstraktna klasa Osoba je osnovni oblik svih učesnika na sistemu i sadrži osnovne informacije za svaku osobu tako da je veoma lako napravit da se ta klasa nadogradi nekom vrstom izvođenja.

Liskov princip zamjene– ono što nas je vodilo prilikom dizajniranja dijagrama klasa svakako se uklapalo u liskov princip zamjene. To se vidi u svim klasama kao npr. klasa Osoba i izvedena klasa Profesor. Možemo primijetiti da se integritet sistema ne bi narušio zamjenom ove dvije klase u hijerarhiji klasa. Isto to vrijedi i za izvedene klase kao što su klase Student, Staratelj, Učenik, itd.

Princip inverzije ovisnosti– Ovaj princip je također zadovoljen. U našem dijagramu se može vidjeti da detalji zavise od apstrakcija a ne obrnuto što se zapravo i jeste ovaj princip. Promjene u sistemu tačnije u nekom interfejsu je promjena na svim mjestima gdje je ona potrebna što dovodi do zaključka da je takve promjene potrebno vršiti na jako malom broju mjesta u sistemu, te je na osnovu toga sistem jako fleksibilan i održiv.

Princip izoliranja interfejsa – Prilikom dizajniranja dijagrama dobro smo pazili na broj metoda unutar interfejsa, tako da se ne bi desilo da neka klasa koja implementira interfejs mora implementirati metode koje ta klasa ne bi koristila. U našem dijagramu ne postoji interfejs kojem broj metoda prelazi broj 4 iz čega se vidi da svaka kalsa implementira optimalan broj metoda interfejsa. Također smo nastojali da čak I za istu klasu odvojimo vise interfejsa da se ne bi desilo da neke klase koje su izvedene ne bi morale koristiti sve interfejse koje se vezuju za tu osnovnu klasu.