Univerzitet u Sarajevu
Elektrotehnički fakultet u Sarajevu
Odsjek za računarstvo i informatiku

ReadTheWorld

SOLID principi

Objektno orijentisana analiza i dizajn

Iris Pjanić
Adna Husičić
Mirza Kadrić

SOLID principi

Single Responsibility Principle

Ovaj princip govori da bi klasa trebala imati samo jedan razlog za promjenu. Ovaj princip je već ispoštovan u prvoj verziji dijagrama, npr.: klasa "Rad" ima odgovornost isključivo za komponente koje su vezane za jedno korisnikovo djelo. Iako ova klasa sadrži listu elemenata tipa Recenzija, postoji klasa "Recenzija" koja upravlja promjenama vezanim za recenziju. Također, klasa Rad se ne bavi drugim funkcionalnostima jer su one raspoređene u nekim drugim klasama. Slično vrijedi i za ostale klase, te je time ovaj princip ispoštovan.

Open - closed Principle

Ovaj princip govori da sistem treba biti otvoren za nadogradnje a zatvoren za promjene. Obzirom na način kako su klase u sistemu implementirane, ovaj princip je od početka izgradnje bio u fokusu jer je bitno da nove funkcionalnosti možemo dodavati na jednostavan način koji ne zahtijeva promjene. Primjer možemo vidjeti u bilo kojoj od klasa koja ima atribut tipa liste elemenata neke druge klase, tako da izmjena jedne klase ne znači da moramo mijenjati ostatak sistema.

Liskov Substitution Principle

Ovaj princip se brine o tome da su sva nasljeđivanja dobro implementirana. U našem slučaju, u sistemu imamo klasu Prijava iz koje su naslijeđene klase: PrijavaRada i PrijavaKorisnika. Svaka od klasa se može koristiti na mjestima gdje se koristi i bazna klasa i njihove modifikacije u odnosu na baznu klasu su minimalne, ali bitne, jer doprinose boljoj organizaciji i specifikaciji problema koji su doveli do same prijave.

Interface Segregation Principle

Kako trenutno u sistemu nemamo niti jedan interfejs, ovaj

princip ćemo pratiti jedino ukoliko se javi potreba za uvođenjem određenog interfejsa, a sada ćemo smatrati da je ispoštovan.

Dependency Inversion Principle
Jednostavna interpretacija ovog principa je da ne treba ovisiti od konkretnih klasa. Pri nasljeđivanju više klasa, bazna klasa treba biti apstraktna. Ovaj princip jeste ispoštovan, međutim jedina promjena koju je trebalo napraviti jeste da se na dijagramu istakne da je klasa Prijava apstraktna (italic). Što se tiče same realizacije, tu nisu potrebne prepravke jer, iako klasa Prijava nema apstraktnih metoda ona je ipak apstraktna klasa jer nema smisla da se instancira prijava bez da naglasimo na koji objekat se ona odnosi.