Model sistema s aspekta SOLID principa

Kao što znamo, SOLID principa ima ukupno 5, te će u nastavku model sistema biti opisan kroz svaki od tih 5 principa sa osvrtom na priloženi dijagram klasa i na način kako su ti principi, kroz klase na dijagramu, zadovoljeni.

1. Princip pojedinačne odgovornosti

Princip pojedinačne odgovornosti (eng. **S**ingle Responsibility Principle, SRP) definiše pravilo koje glasi: „**Klasa bi trebala imati samo jedan razlog za promjenu!**“. U suštini, ovim pravilom se nalaže da jedna klasa treba imati samo jednu odgovornost. Model sistema, koji je predstavljen priloženim dijagramom klasa, ispunjava zahtjeve ovog principa. Svaka od klasa, koje sačinjavanju model sistema, bazirana je na jedinstvenoj odgovornosti. Za većinu klasa koje su predstavljene na dijagramu, ispunjenje ovog principa vidljivo je na prvi pogled, tako da će model sistema s aspekta ovog principa biti opisan uz pomoć klase kod koje nije pretjerano očigledna prisutnost ovog principa. Naime, klasa *Utakmica* bila je zamišljena na način da enkapsulira neke bitne informacije vezane za održavanje utakmice. Dvije funkcionalnosti sistema, koje su vezane za utakmice, svakako su gledanje utakmice putem prijenosa uživo, ali i kupovina ulaznica za utakmice. Realizacija te dvije funkcionalnosti, s aspekta modela sistema, mogla bi se klasificirati kao jedna odgovornost, a to je gledanje utakmice. Međutim, klasa *Utakmica* ne bi ispunjavala SRP princip ukoliko bi, pored čuvanja bitnih informacija vezanih za utakmicu, čuvala i neke elemente (atribute ili metode) koji se odnose na događaj gledanja utakmice. Razlog je jednostavan, čuvanje informacija o utakmici, te s druge strane gledanje utakmice, ne mogu se klasificirati kao jedna odgovornost jer su po prirodi to dva različita procesa. U skladu sa navedenim, formirana je nova klasa koja je nazvana *GledanjeUtakmice*, a kojom su se te dvije navedene odgovornosti veoma lako rasporedile u dvije klase.

2. Otvoreno zatvoreni princip

Otvoreno zatvoreni princip (eng. **O**pen Closed Principle, OCP) definiše pravilo koje glasi: „**Entiteti softvera (klase, metode, moduli) trebali bi biti otvoreni za nadogradnju, a zatvoreni za promjenu!**“. Model sistema, koji je predstavljen priloženim dijagramom klasa, odražava sistem s aspekta svih funkcionalnosti čija je egzistencija zamišljena. Kako u ovoj fazi modeliranja još uvijek nisu u potpunosti definisane sve metode koje će biti prisutne u klasama, i dalje govorimo o prvoj verziji dijagrama klasa kao temelju na kojem će biti zasnovan ostatak modeliranja. Međutim, taj temelj u potpunosti obuhvata svaku osobinu sistema koji se gradi, tako da u budućnosti ne može doći do izmjene nekih od stavki ovog dijagrama nego eventualno do nadogradnje.

3. Liskov princip zamjene

Liskov princip zamjene (eng. **L**iskov Substitution Principle, LSP) definiše pravilo koje koje glasi: „**Podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima!**“. Na priloženom dijagramu klasa prisutne su dvije pojave nasljeđivanja. U prvom slučaju, klasa *Korisnik* je bazna klasa, a njezina djeca su klasa *RegistrovaniKorisnik* i klasa *VIPKorisnik*. Veoma je jednostavno

uočiti da je LSP princip u ovom slučaju zadovoljen jer je svaki registrovani korisnik, kao i svaki VIP korisnik, ujedno i obični korisnik ovog sistema. U drugom slučaju, klasa *Zaposlenik* je bazna klasa, a njezina djeca su klasa *Administrator* i klasa *SportskiUrednik*. I u ovom slučaju je veoma lako na intuitivan način shvatiti prisutnost LSP principa jer su i administrator i sportski urednik zaposlenici sistema.

4. Princip izoliranja interfejsa

Princip izoliranja interfejsa (eng. **I**nterface Segregation Principle, ISP) definiše pravilo koje glasi: „**Klijenti ne trebaju ovisiti o metodama koje neće koristiti!**“. Prilikom kreiranja modela sistema u vidu prve verzije dijagrama klasa, nije se javila potreba za korištenjem interfejsa tako da ovaj princip svakako nije narušen.

5. Princip inverzije ovisnosti

Princip inverzije ovisnosti (eng. **D**ependency Inversion Principle, DIP) definiše pravilo koje glasi: „**Moduli visokog nivoa ne bi trebali ovisiti o modulima niskog nivoa!**“. Najjednostavnije rečeno, ovim principom se nalaže da konkretne klase više zavise od apstraktnih klasa ili od interfejsa, nego od nekih drugih konkretnih klasa. Ispunjenje ovog principa je realizaovano uvođenjem dvije apstraktne klase u model sistema, a to su klasa *Korisnik* i klasa *Zaposlenik* čime je ostvaren efekat da se na početku hijerarhija kod tih nasljeđivanja nalaze apstraktni elementi koji u budućnosti neće biti mijenjani.