Analiza primijenjenih SOLID principa

Napomena: dijagram klase ne sadrži gettere/settere I sve konstruktore koji se mogu javiti čisto zbog preglednosti.

- Princip S: Svaka klasa treba imati samo jednu ulogu

U našem projektu, pokušali smo da jedna klasa ima samo jednu odgovornost: klasa Korisnik je klasa iz koje su izvedeni korisnici aplikacije,dakle korisnik “FizickoLice”,”Ustanova”,”Admin”,VIPKorisnik”,te ta klasa se brine o osnovnim podacima koji se tiču korisnika I njihovim izmjenama. Dakle,te sve klase se brinu o dodavanju I modifikaciji objekata od kojih se klase sastoje I isčitavanjem tih vrijednosti.Dalje,klasa koja bi potencijalno mogla da bude faktor za rušenjem ovog pricipa je klasa Ustanova,koja može da komunicira sa BazomPodataka,međutim, operacije koje služe ze pristup bazi I čitanje iz baze su izdvojene u posebnu klasu BazaPodataka,te smo na taj način pokušale riješiti taj problem. Dalje,klase koje imaju mogućnost da printaju odgovarajuće izvještaje,omogućeno je da te akcije budu izdvojene u zasebne interfejse. Klasa Događaj se brine o svojim atributima,njihovim modificiranjem I dodavanjem,baš kao I klasa Rezervacija.Klasa Obavijest je poseban vid klase koja nema mnogo atributa I samo služi da na jednom mjestu imamo templejte obavještenja I da kroz svoje metode omogući da obavještenja se upisuju u liste obavještenja odgovarajućih korisnika.Klasa Obavijest služi za implementaciju metoda iz interfejsa za obavještavanje.Klasa Admin ima odgovornosti vezane za odobravanje događaja,zahtjeva,kreiranja računa,a za te akcije postoji ideja da se kroz ove metode utvrdi da li je događaj ili zahtjev validan da se pojavi u aplikaciji,te da u tim metodama admin pošalje obavještenje odgovarajućem korisniku. Uz admin,postoji I interfejs koji služi za odobravanje zahtijeva. Klasa Transakcija,nam služi da uplatimo rezervaciju I da se brine o osnovim podacima transakcije. Također,uz ovu klasu,imamo I interfejs za transakciju odnosno plaćanje. Najveća klasa u sistemu je klasa EventManager,koja komunicira najviše sa bazom podataka,daje odobrenja za zahtjeve,nudi mogućnost pravljenja izvještaja. Dakle,sve što je potencijalno van uloge jedne klase,izdvojeno je u odgovarajuće interfejse,da bi nam bilo lakše poslije kod nadogradnje programa.

- Princip O: Klasa treba biti otvorena za nadogradnje,ali zatvorena za modifikacije

Što se tiče ovog principa, klase koriste druge klase,ali u većini metoda koriste se postupci za ubacivanje objekata u liste,što je generička operacija. Također,za pojedine klase,postoje enumerativni tipovi koji omogućavaju da se metode ne moraju mijenjati,ukoliko se kreira novi tip,I da ukoliko se kreira neki novi tip,npr.novi tip karte,dovoljno je samo dodati novu jednu metodu u klasu Rezervacije,koja će da ubuduće dati mogućnosti odabira te vrste karte pri rezervaciji.Dakle,dešavat će se samo nadogradnja I ponuda nove mogućnosti.

- Princip L: Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima bez da to utječe na ispravnost rada programa

Na našem primjeru, na mjestima gdje imamo pravljenje arhivskih listi ili nekih listi korisnika, korisnik koji je fizičko lice je također korisnik,baš kao I ustanova I admin, te sasvim je uredu da u jednoj metodi koja dodaje korisnike u listu napišemo da je to lista tipa Korisnik I to će imati smisla. Također,klasa koja se bavi obavještavanjem korisnika ima atribut primaocObavjestenja koji je tipa Korisnik. Na ovom mjestu,sasvim je svejedno da li obavještavamo fizičko lice,ustanovu ili admina,u pitanju je korisnik.

- Princip I: Bolje je imati više specifičnih interfejsa, nego jedan generalizovani

Da bismo,što je više moguće, ispoštovale princip S kod klasa,uvedeno je nekoliko interfejsa koji bi trebali da olakšaju posao,tako da postoje interfejsi koji omogućavaju kreiranje izvještaja u zavisnosti od potrebe klase I njenih osobina (neće isti izvještaj biti o uplati VIP članarine I izvještaj koji služi ustanovi da ima uvid u sve transakcije I rezervacije u jednom mjesecu),postoje izvještaji koji treba da pomognu klasama kod davanja odobrenja,izračuna transakcija,članarina,validacije podataka I obavještavanja.Dakle,ti interfejsi nam služe kao mini podprogrami koje će da koriste klase.

- Princip D: Sistem klasa I njegovo funkcionisanje treba ovisiti o apstrakcijama,a ne o konkretnim implementacijama

Ovaj princip zahtijeva da pri naslijeđivanju od strane više klasa bazna klasa bude uvijek apstraktna,što je kod nas I slučaj-klasa korisnik je apstraktna klasa.