SOLID principi

S princip

Klasa „Korisnik narušava princip S, jer princip S naglašava da klase trebaju da znaju o samo jednoj stvari. „Korisnik“ ima mogućnost da upiše kurs, ostavi ocjenu, ostavi komentar, započne kviz iz date lekcije i kviz iz kursa koji pohađa. Ovaj princip bi se mogao poštovati, ukoliko se naprave interfejsi: interfejs za upisivanje na kurs, za ostavljanje ocjene i za kvizove pojedinačno (kurs i lekcija), i svaki interfejs bi radio jednu stvar. Kako su klase „KorisnikSaFakultetskimMailom“ i „KorisnikBezFakultetskogMaila“ naslijeđene klase iz klase „Korisnik“, i za njih važi da se ne pridržavaju S principa. Još one imaju dodatnu metodu koja bi omogućila korisnicima sa/bez e-maila da se odjave, za šta bi trebao još jedan, dodatni interfejs. Također, ista situacija vrijedi i za klasu „Administrator“ s obzirom da je naslijeđena klasa iz klase „Korisnik“, ali ima dodatnu mogućnost prikaza statistike. Ova klasa bi se mogla pridržavati S principa ukoliko se naprave, pored nabrojanih interfejsa, i interfejs za pregled statistike. Klasa „Kurs“ također ne zadovoljava S princip zbog toga što klasa sadrži metode kao što su: „upisiKurs“, „dodajLekciju“, „dajLjestvicu“. Klasa zna za više stvari, tako da je ovaj princip narušen. S princip bi se poštovao ukoliko napravimo tri interfejsa za tri različite stvari: upis korisnika na kurs, dodati lekciju za kurs i dati ljestvicu. Klasa kao što je „Posjetilac“ ne narušava ovaj princip iz razloga što posjetilac ima mogućnost registracije i prijave, a s obzirom da to predstavlja jednu logičku cjelinu, ovaj princip nije narušen, te klasa zna o samo jednoj stvari: računu korisnika. Da li će to biti prijava ili registracija, ovisi o posjetiocu naše aplikacije. „Lekcija“ se pridržava S principa jer ima samo ulogu prikaza lekcije korisniku. „Oblast“ se također pridržava ovog principa s obzirom da ima metodu „dodajKurs“. Ovim smo postigli da u jednoj oblasti je moguće samo dodavati kurseve, s obzirom da se oblast sastoji od više kurseva. Klasa „Kviz“ i naslijeđene klase iz nje, kao što su „KvizZaKurs“ i „KvizZaOblast“ ne narušavaju S princip jer rade samo jednu stvar: mogućnost predaje tog kviza i mogućnost prikaza broja tačno odgovorenih pitanja. Ovo klase koje su naslijeđene iz apstraktne klase „Kviz“ smo odvojili na ovaj način kako bismo imali pregledniji uvid u odrađene kvizove koje korisnik uradi. Imat ćemo odvojene poene za ove kvizove što omogućava bolju evidenciju. „Pitanje“ i „Ljestvica“ se pridržavaju principa, respektivno, klasa provjerava da li je pitanje tačno odgovoreno i ljestvica ima mogućnost osvježavanja podataka s obzirom da svaki dan različiti korisnici pristupaju kvizu, te se rang ljestvica mijenja.

Klase koje zadovoljavaju S princip bit će jednostavnije za implementirati i preglednije za osobu koja pregleda našu dokumentaciju. Očekujemo da one klase koje zadovoljavaju navedeni princip imaju jasno definisane „poslove“ koje obavljaju s obzirom da te klase znaju za samo jednu stvar koju rade.

O princip

Nijedna klasa ne narušava ovaj princip. Jedina klasa koja bi iole mogla narušavati O princip je „Ljestvica“, ali kako atribut kojeg ona sadrži „kurs“ koji je tipa „Kurs“ služi za identifikaciju odabranog kursa, a ne za njegovu modifikaciju, ova klasa ga ne narušava.

Također, promjena klase „Kurs“ nikako ne utiče na klasu „Ljestvica“, to jeste možemo mijenjati kurs, a da ljestvica ostane ista i da opet funkcioniše kako treba.
Dakle, O princip zadovoljavaju sve klase što nam osigurava da ni u jednoj klasi nećemo imati mogućnost da izvornu klasu na bilo koji način promjenimo. Ovo je jako bitno iz razloga što vrlo uz dobru implementaciju, neće doći do bilo kakve konfuzije među klasama. Svaka obavlja svoj zadatak(e) i ne može mijenjati ni na koji način neku drugu klasu. To je naše očekivanje od ovog principa. Olakšava bezbjednu „komunikaciju“ među svim nabrojanim klasama.

L princip

U našem konkretnom primjeru, iz klase „Korisnik“ su naslijeđene tri klase: „KorisnikSaFakultetskimMailom“, „KorisnikBezFakultetskogMaila“ i „Administrator“. L princip zahtijeva da podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima. „Administrator“ nije zamjenjiv osnovnim tipom, zbog mogućnosti pregleda statistike, što „Korisnik“ nije u mogućnosti, te ovaj princip nije zadovoljen. „KorisnikSaFakultetskiMailom“ i „KorisnikBezFakultetskogMaila“ zadovoljavaju ove principe s obzirom da ih je moguće zamijeniti sa njihovimo osnovnim tipom, tj. „Korisnikom“. Naslijeđene klase iz klase „Kviz“ su „KvizZaLekciju“ i „KvizZaKurs“. Obje naslijeđene klase je moguće zamijeniti sa klasom „Kviz“, te je princip zadovoljen kod njih.

Očekujemo da će ispunjenje ovog principa kod klasa koje ga ispunjavaju, olakšati implementaciju, te neće biti nikakvih problema pri nadograđivanju postojećeg koda s obzirom na zamjenu podtipova i osnovnih tipova.

I princip

Trenutno nemamo nikakav interfejs, tako da nemamo interfejs koji bi narušavao ovaj princip.

Kao što smo naglasili u prvom principu, tj. S principu, ukoliko bismo imali odgovarajuće interfejse koji su navedeni u tom pasusu, ne bismo imali kršenje S principa, ali s obzirom da nemamo nikakvog interfejsa, I princip ne može biti narušen.

D princip

U našem primjeru, jedina nasljeđivanja imamo iz klasa „Korisnik“ i klase „Kviz“. Obje ove klase su apstraktne, a kako u ovom principu funkcionisanje sistema treba ovisiti o apstrakcijama, što je ovdje i slučaj, ovaj princip je ispunjen. Također, princip se može i jednostavnije interpretirati, a to je da ne treba ovisiti od konkretnih klasa, što je ovdje zadovoljeno.
Očekujemo od ispunjenja ovog principa efikasniju primjenu baznih klasa s obzirom da se interfejsi i apstraktne klase mnogo manje mijenjaju s obzirom na njihove konkretne izvodnice. Slijedeći D princip smanjuje se uticaj koji promjena može imati na sistem.