Univerzitet u Sarajevu
Elektrotehnički fakultet
Objektno orijentisana analiza i dizajn

SOLID PRINCIPI

1. Single-Responsibility Principle

Ovaj princip je princip pojedinačne odgovornosti klasa. Svaka klasa je odgovorna samo za ono što se tiče direktno nje. Klasa Klijent je odgovorna za informacije o klijentu. Klasa Zahtjev posjeduje instancu klase Klijent, onog klijenta koji je podnio zahtjev. Tip zahtjeva je enum, može biti poliranje, popravak_vezova i iznajmljivanje. Svaki zahtjev može imati više stavki klase Oprema. Oprema je apstraktna klasa, nju nasljeđuju klase Skije, Pancerice, Štapovi, Snowboard, SnowboardCipele i Kaciga. U ovim klasama različite vrste opreme imaju različite atribute, a klasa Oprema sadrži samo ono što je zajedničko svim klasama koje su iz nje naslijeđene i radi samo ono za šta ona može biti zadužena. Klase Uposlenik i Vlasnik su naslijeđene iz apstraktne klase Osoblje. Naprimjer da smo klasu Vlasnik naslijedili iz klase Uposlenik, ona bi onda vjerovatno "znala previše", tj. sadržavala bi i metode koje nikada ne bi pozivala, jer je zamišljeno da vlasnik i uposlenik obavljaju drukčije dužnosti. Tako vodimo računa principu pojedinačne odgovornosti.

1. Open-Closed Principle

Po ovom principu, entiteti softvera (klase, moduli, metode) trebaju biti otvoreni za nadogradnju, ali zatvoreni za modifikacije. Sve klase koje su naslijeđene iz klase Oprema, mogu se izmijeniti bez potrebe za izmjenom osnovne klase. Kao i klase Uposlenik i Vlasnik koje su naslijeđene iz klase Osoblje. Možemo im dodavati nove metode, atribute bez promjene klase iz koje su naslijeđene. Možemo ih mijenjati, bez promjene svrhe njihovog postojanja.

2. Liskov Substitution Principle

Po ovom principu, podtipovi moraju moći biti zamijenjeni svojim osnovnim tipovima tako da i dalje program ispravno funkcionira. Klase Skije, Pancerice, Štapovi, Snowboard, SnowboardCipele i Kaciga mogu zamijeniti objekte klase Oprema te Klase Uposlenik i Vlasnik mogu zamijeniti objekte klase Osoblje.

3. Interface Segregation Principle

Ovaj princip nalaže da klijenti ne treba da ovise o metoda koje neće upotrebljavati. Baš zato u našem slučaju postoje odvojene klase Uposlenik i Vlasnik, koje imaju zajedničke neke metode i atribute, no imaju i raličite metode. Obični uposlenik ne može koristiti metodu promijeniCijenu(), i on o njoj ne treba da ovisi. U slučaju da želimo imati dvije vrste klijenta, implementirali bismo interfejs Klijent, te iz njega naslijedili klase KlijentOnline i KlijentUživo, ako bismo u budućnosti željeli uvesti dvije vrste plaćanja i opet držati se pravila ovog principa.

5. Dependency Inversion Principle

Ovaj princip nalaže da moduli visokog nivoa ne bi trebali ovisiti od modula niskog nivoa. Klasa StavkaNarudžbe ovisi direktno od apstraktne klase Oprema. Time klasa Zahtjev može sadržavati više različitih stavki klase Oprema, svejedno koje konkretno klase. Postignuto je da su moduli ovisni od apstrakcija.