# Merch store - SOLID principi

Single responsibility i open-closed principi:
Klasa Korisnik: Klasa koja opisuje korisnika. Implementira 2 interfejsa: INarucivanje i IPlacanje. Naručivanje kao funkcionalnost je odvojeno od ove klase iz razloga što narucivanje mijenja stanje baze (tj. nije zadatak ove klase), a u budućnosti bi se mogao dodati novi tip korisnika koji ne bi mogao naručivati (npr. guest). Kontrola uplatama nije stvar sa kojom bi se ova klasa trebala baviti, pa je taj dio odvojen u zaseban interfejs.

Klasa Administrator: Klasa koja opisuje administratora. Implementira 4 interfejsa: IObavijest, ITerminiranje, IPotvrdivanje, IBaza. Razlog što su ove funkcionalnosti administratora odvojene od same klase je zbog postojanja mogućnosti nadogradnje sistema npr. mogućnost dodavanja moderatora koji će moći potvrđivati narudžbe ali ne i terminirati korisnike ili mogućnost da jedan korisnik šalje poruku drugom korisniku.

Klasa Garderoba: Klasa koja opisuje garderobu koju korisnik može dizajnirati. Klasa nema specifičnih zahtijeva. Klasa MerchStore: Klasa koja opisuje merch store. Slično kao i klasa Garderoba, implementira interfejs IMerchStoreBaza.

Klasa NarudzbaDizajnera: Klasa koja opisuje stavke narudzbe koje je korisnik postavio pri narudžbi. Ova klasa nema specijalnih zahtijeva.

Jedan dodatni primjer ispunjavanja Single responsibilty principa su interfejsi INarucivanje i IDizajnNarucivanje koji zasebno služe za naručivanje od strane dizajnera i od strane kupca.

Liskov substitution princip:
Ovaj princip je primjenjen na 2 načina.
Prvi način je vezan za klase Administrator i Korisnik. Vidimo da su atributi ovih klasa slični te da bi možda mogli postaviti da je Korisnik generalizacija Administratora. Međutim to nije dobra ideja iz razloga što su korisnik i administrator imaju različite prioritete i mogućnosti (npr. administrator nema mogućnost naručivanja ili dizajniranja).
Drugi način je vezan za klase Garderoba i NarudzbaDizajnera. Slično kao i prethodni način, ove klase imaju slične atribute, međutim one se koriste u potpuno drugačijim situacijama, te ih ne treba povezivati generalizacijom.

Interface segregation princip:
Prva primjena je kod interfejsa koje implementira klasa Administrator. Naime mi smo mogli interfejse ITerminiranje i IPotvrdivanje spojiti u jedan interfejs (npr. IAdministratorPrivilegije), ali onda ukoliko želimo dodati novi tip administratora (moderator) koji bi samo mogao potvrdivati narudzbe ali ne i terminirati, to ne bismo mogli uraditi.
Druga primjena je kod interfejsa koje implementira klasa Korisnik. Slično kao i prethodna primjena, mi smo mogli ove interfejse spojiti u jedan interfejs INarucivanje, ali ako npr. želimo dodati mogućnost da neregistrovani korisnici mogu samo naručivati (ali ne i dizajnirati i posjedovati merch store), to ne bi mogli.

Dependency inversion princip:
Primjena kod klase MerchStore. Ukoliko želimo dodati novu klasu npr. VIPMerchStore, tada ukoliko korisnik ima atribut tipa MerchStore, on neće moći posjedovati VIPMerchStore. Stoga smo napravili novu apstraktnu klasu Store, te promjenili atribut kod klase Korisnik u Store.