SOLID principi

- **S** princip je zadovoljen za sve klase, jer većina klasa posjeduje od metoda samo getere i setere. Ostale klase pored getera i setera vrše samo dodavanje ili brisanje članova iz liste koju posjeduje kao atribut, tako da je ovaj princip zadovoljen.
- **O** princip je zadovoljen za sve klase. Što se tiče narušavanja ovog principa, klase koje bi potencialno predstavljale problem su one klase koje kao atribute sadrže druge klase. Međutim, naš sistem je baziran na tome da sve naše klase pretežno imaju metode getera i setera, te prilikom uvođenja novih funkcionalnosti neće biti potrebe za bilo kakvom modifikacijom već postojećih klasa.
- **L** princip nije zadovoljen, jer tokom analize smo zaključili da ne možemo na svim mjestima gdje se koristi osnovni objekat koristiti i izvedeni objekat. Problem je u klasi Zahtjev, koja ima atribut podnosilacZahtjeva koji je tipa Korisnik. Problem je u tome što ne možemo zamijeniti Korisnik sa klasom Uprava, a Uprava je izvedeni tip, jer nema smisla da Uprava popunjava zahtjev za upis, cimeraj... Problem je riješen na taj način da iz klase Zahtjev izvodimo prvo klase ZahtjevStudenta i ZahtjevRestorana, koje su opet apstraktne. Atribut podnosilacZahtjeva je uklonjen iz klase Zahtjev, a dodan u klase ZahtjevStudenta i ZahtjevRestorana, s tim da je u klasi ZahtjevStudenta tipa Student, dok je u klasi ZahtjevRestorana tipa Restoran. Iz klase ZahtjevStudenta se izvode klase ZahtjevZaUpis, ZahtjevZaPremjestanje i ZahtjevZaCimeraj. Iz klase ZahtjevRestorana je izvedena klasa ZahtjevZaNabavkuNamirnica. Ovime smo riješili problem koji smo imali, jer sada u Zahtjevu nemamo atribut tipa Korisnik, koji je pravio problem. Klasa Korisnik zadovoljava ova princip.
- **I** princip je zaadovoljen imamo samo dva interfejsa koji obavljaju samo jednu vrstu akcija. Konkretno interfejs AzurirajStanjeBonova, ažurira stanje bonova, a PregledStanjaBonova, vrši samo pregled stanja bonova.
- **D** princip je zadovoljen, jer sve bazne klase koje smo koristili u apstraktne.