SOLID PRINCIPI

1. Princip pojedinačne odgovornosti (Single responsibility Principle)

- Ovaj princip nije zadovoljen zbog činjenice da svaka klasa treba da ima pojedinačnu odgovornost. Međutim, naše klase KorisnikSaNalogom i KorisnikSistema rade neke aktivnosti koje su inače vezane za samu bazu. Taj problem ćemo riješiti tako što ćemo dodati interfejs IBaza koji ima default–ne implementacije metoda koje će onda naše klasa koristiti.

2. Otvoreno-zatvoreni princip (Open/Closed Principle)

- Ovaj princip kaže da svaka klasa mora biti otvorena za nadogradnje i zatvorena za modifikacije. Ukoliko promijenimo jednu klasu neće doći do promjene druge klase.Ni u jednoj klasi nemamo omogućeno mijenjanje druge klase. Potencijalno bismo mogli u klasi Iznajmljivanje izdvojiti popust kao zaseban interfejs.

3. Liskov princip zamjene (Liskov Substitution Principle)

- Ovaj princip nalaže da bilo koja upotreba bazne klase omogućava upotrebu i izvedenih klasa, sa istim rezultatom. Podtipovi moraju biti zamjenjivi njihovim osnovnim tipovima.Klasa KorisnikSaNalogom je nasljeđena iz klase KorisnikSistema a ovaj princip nije zadovoljen jer klasa KorisnikSaNalogom sadrži neke metode koje uopšte nisu veza za izvednu klasu KorisnikSistema. Nasljeđivanje u ovom slučaju nema smisla pa će se princip zadovoljiti tako što ćemo ove dvije klase napraviti kao dvije zasebne: Admin i KorisnikSaNalogom.

4. Princip izoliranja interfejsa (Interface Segregation Principle)

- Imamo samo jedan interfejs IBaza i on omogućava klasama da koriste njegove default–ne metode, samim tim je ovaj princip zadovoljen.

5. Princip inverzije ovisnosti (Dependency Inversion Principle)

- Sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama. Kako u našem dijagramu (u kojem su gore navedene ispravke) nemamo nasljeđivanje ovaj princip je zadovoljen.