1. S-Single Responsibility Principle (Princip pojedinačne odgovornosti)

Ovaj princip je ispoštovan jer sve klase imaju pojedinačne

odgovornosti, odnosno svaki posao se veže za jednu klasu.

Ako posmatramo klasu Korisnik, možemo primijetiti da ona posjeduje samo

one atribute koje svaki korisnik mora imati, te će klasa posjedovati

samo one metode koje će moći vraćati i modifikovati te atribute.

Očito je da ova klasa poznaje samo jednu stvar (korisnika) jer sadrži

samo atribute i metode koji su usko vezani uz samog korisnika.

Izmjena ove klase može izazvati samo potreba za promjenom Korisnika.

2. O-Open/Closed Principle (Princip otvorenosti/zatvorenosti):

Ovaj princip je ispostovan u primjeru klasa Korisnik i klasa koje su naslijeđene iz nje. Klasu “Kupac” mogli bismo dodati bez da modifikujemo postojeći kod.

Klase su otvorene za proširenje, ali zatvorene za modifikaciju. Također, ako bismo željeli dodati novu vrstu korisnika poput "Korisnik", trebalo bi moći dodati novu klasu “UposlenikUsluznogObjekta” bez mijenjanja postojećeg koda u klasama "Korisnik" ili "Narudzba".

3. L-Liskov Substitution Principle (Princip zamjene Liskov):

Svaka podklasa trebalo bi da bude zamjenjiva s njihovom baznom klasom bez narušavanja ispravnosti programa. To znači da bi "Kurir", "Administrator" i "Korisnik" trebali biti zamjenjivi gdje god se očekuje instanca klase "Korisnik".

Za primjer možemo da uzmemo situaciju u kojoj, prilikom izmjene username-a ili nekih drugih korisničkih podataka, oni se mogu promijeniti za instance svih naslijedjenih klasa, bez narušavanja ispravnosti programa i ponavljanja koda. Naravno, ovo važi uz pretpostavku da je takva funkcionalnost vezana isključivo za baznu klasu.

4. I-Interface Segregation Principle (Princip izoliranja interfejsa)

U našem primjeru imamo 3 vrste korisnika: kupac, kurir, administrator, pri čemu svaki od njih obavlja svoje funkcije. Na primjer, kupac ne može da se miješa u promjenu cijena bilo kojih proizvoda, pa to zahtjeva posebne interfejse, čime je ovaj princip zadovoljen.

5. Dependency Inversion Principle (Princip inverzije ovisnosti)

Klasa Korisnik bi trebalo da posjeduje interfejs u kom će postojati metode koje su zajedničke za klase Administrator, Kupac i Kurir pa je time ispoštovan ovaj princip. Na primjer, klasa Korisnik sadrži metodu izvrsi_placanje, a klase Administrator, Kupac i Kurir zatim nasljeđuju Korisnik klasu i implementiraju tu metodu na svoj način.