**Task 4 – Solid principi**

**1.Princip pojedinačne odgovornosti**

Princip pojedinačne odgovornosti ( eng. Single Responsibility Principle-SRP) glasi:

KLASA BI TREBALA IMATI SAMO JEDAN RAZLOG ZA PROMJENU.

Ovaj princip je, kao u svakom dobro dizajniranom sistemu, ispoštovan u skoro svim klasama, a postignut je dobrim raspoređivanjem poslova po klasama, tako da svaka klasa ima samo jedan razlog za promjenu.

Za primjer ćemo uzeti klasu Plaćanje, kojoj će jedini razlog za promjenu biti promjena rezervacije, koja će se odraziti na račun koji generiše navedena klasa. Kada bi klasa Plaćanje bila zadužena za više poslova, npr. kada bi, pored generisanja računa, radila i generisanje ugovora, validacije vezane za rezervaciju, te validacije vezane za ponudu, skoro svaka promjena u sistemu bi uzrokovala promjenu u klasi. Zato su navedeni poslovi podijeljeni na više različitih klasa.

**2.Otvoreno zatvoren princip**

Otvoreno zatvoren princip ( eng. Open Closed Principle-OCP) glasi:

ENTITETI SOFTVERA (KLASE, MODULI, FUNKCIJE) TREBALI BI BITI OTVORENI ZA NADOGRADNJU, ALI ZATVORENI ZA MODIFIKACIJE.

Ako posmatramo klase Ponuda i StavkaRezervacije uočit ćemo da je ovaj princip zadovoljen.

Npr, da su direktno uspostavljene ovisnosti ovih klasa sa bazom podataka ovaj princip bi bio narušen. Nadogradnja, manipulacija, modifikacija baze podataka imala bi direktan utjecaj na spomenute klase u vidu modifikacije istih.

Uspostavljanje ovisnosti ovih klasa sa interfejsom baze podataka ovaj princip je zadovoljen.

3. **Liskov princip zamjene**

Liskov princip zamjene (eng. Liskov Substitution Principle-LSP) glasi:

PODTIPOVI MORAJU BITI ZAMJENJIVI NJIHOVIM OSNOVNIM TIPOVIMA.

Kako je ovaj princip ispoštovan, moguće je vidjeti na primjeru klase Klijent, te klasa izvedenih iz nje, Povlašteni, Nepovlašteni I Lider. Naime, klasa Klijent sa svim svojim atributima i metodama u potpunosti opisuje svoje izvedene klase, te bi se one mogle zamijeniti svojom baznom klasom, ukoliko bi za tim bilo potrebe. Međutim, da je klasa Klijent, uz svoje metode, imala I metodu vratiPopust ili metodu validirajKreditnuKarticu, sve njene izvedene klase se ne bi mogle sa potpunom tačnošću zamijeniti baznom klasom (nepovlašteni korisnik nema popust, iako bi se to moglo riješiti vraćanjem nule, ali metoda validirajKreditnuKarticu ne bi smjela imati veze sa Nepovlaštenim I Povlaštenim klijentom jer plaćati može samo lider tima).

**4.Princip inverzije ovisnosti**

Princip inverzije ovisnosti (eng. Dependency Inversion Principle-DIP) glasi:

**A.** MODULI VISOKOG NIVOA NE BI TREBALI OVISITI OD MODULA NISKOG NIVOA. OBA BI TREBALO DA OVISE OD APSTRAKCIJA.

**B.** MODULI NE BI TREBALI OVISITI OD DETALJA. DETALJI BI TREBALI BITI OVISNI OD APSTRAKCIJA.

U biti, trebalo bi se što više nasljeđivati iz apstraktnih klasa ili interfejsa, što je u ovom sistemu u većem dijelu ispoštovano. Na primjeru klasa uključenih u Ponudu (Oprema, Vozilo, Osoblje, Oružje, Hrana) možemo vidjeti da su naslijeđene iz apstraktne klase Komponenta. Međutim, kod klase Klijent ovaj princip nije ispoštovan (iz navedene klase su naslijeđene tri klase, pri čemu ova klasa nije apstraktna), ali nije ni potrebno da klasa Klijent bude apstraktna, jer bi onda neka njena metoda morala biti apstraktna, ili ne bismo mogli sve da implementiramo, pa bismo u svakoj njenoj izvedenoj klasi morali implementirati apstraktnu metodu, iako bi ona radila isto.

**5.Princip izoliranja interfejsa**

Princip izoliranja interfejsa ( eng. Interface Segregation Principle-ISP) glasi:

KLIJENTI NE TREBA DA OVISE O METODAMA KOJE NEĆE UPOTREBLJAVATI.

Ovaj princip je ispoštovan korištenjem interfejsa nad klasom Plaćanje.