SOLID principi

1. **S – *Single Responsibility Principle – SRP (Princip pojedinačne odgovornosti)***

    - Klasa bi trebala imati samo jedan razlog za promjenu.
    - Klase su takve da obavljaju jednu funkciju. Klasa **Osoba** je klasa za aktere sistema, čuva informacije o istim i potrebne stvari vezane za profile. Klasa **Proizvod** koja sadrži podatke o proizvodima i klase **Hrana**, **Piće** koje proširuju proizvode bez miješanja odgovornosti. Slično važi za ostale klase.
    - Svaka klasa treba da ima jednu i samo jednu odgovornost, što je u našem dizajnu klasa ispunjeno, samim tim ovaj princip je ispunjen.

2. **O - Open Closed Principle – OCP (Otvoreno – zatvoreni princip)**

    - Entiteti softvera bi trebali biti otvoreni za nadogradnju, ali zatvoreni za modifikacije.
    - Klasa **Proizvod** je proširena kroz klase **Hrana** i **Pice**, ukoliko poželimo dodati još vrsta proizvoda isto možemo uraditi bez mijenjanja logike klase **Proizvod**.
    - Enum klase su stabilne, a moguća je nadogradnja istih bez mijenjanja logike.
    - Princip je ispoštovan jer je moguće dodavati nove funkcionalnosti i izbore bez mijenjanja postojećih klasa.

3. **L - Liskov Substitution Principle – LSP (Liskov princip zamjene)**

    - Podtipovi moraju biti zamjenjivi nihovim osnovnim tipovima.
    - U našem sistemu klase **Hrana** i **Pice** poštuju ovaj princip jer se mogu koristiti gdje god se koristi klasa iz koje su izvedene tj. **Proizvod**. Ukoliko bi se ijedna od dvije navedene klase ponašala drugačije u budućnosti (npr. ne bi imala mogućnost ostavljanja recenzije) tada bi ovaj princip bio narušen, i ista klasa bi se morala izdvojiti kao zasebna, a ne nasljeđena iz klase **Proizvod**.
    - S obzirom na navedeni primjer, i treći princip je ispoštovan.

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

    - Klijenti ne treba da ovise od interfejsa sa metodama koje neće upotrebljavati.
    - Trenutno u našem dizajnu ne koristimo interfejs s obzirom da je struktura klasa dovoljno razdvojena, pa nijedna klasa nema obavezu implementirati stvari koje joj ne trebaju.
    - Također, imamo posebnu klasu **Obavijest**, kako bi se izbjeglo pretrpavanje korisnika i radnika (klasa O**soba**) bespotrebnim funkcionalnostima.
    - Na ovaj način, princip ISP je također ispoštovan.

5. D - Dependency Inversion Principle – DIP (Princip inverzije ovisnosti)

- Konkretne klase ne bi trebale da ovise od konkretnih klasa, već od apstrakcija.
- U našem projektu, svi akteri sistema će biti izvedeni iz klase **Osoba**, te smo na taj način postigle da sistem ne bude osjetljiv na promjene.
- Funkcionalnost slanja obavijesti korisniku i izvršavanje plaćanja nisu direktno ugrađene u klase kao što su **Narudzba** ili **Radnik**, već se pozivaju kao posebne logičke cjeline. Tako se omogućava da se način izvršenja ovih funkcionalnosti može mijenjati ili proširivati bez potrebe za izmjenom osnovnih klasa koje ih koriste.
- Ovim je ispoštovan i peti princip.