SOLID PRINCIPI

- **Princip S** – U našoj aplikaciji imamo jasno definisane klase koje obavljaju samo jednu vrstu zadatka. Na primjer, klasa Korisnik obavlja funkcije vezane za korisničke račune, dok klasa Vozilo upravlja podacima o vozilima. Klasa Korpa upravlja stavkama u korpi i izračunava ukupnu cijenu, a klasa Narudzba upravlja informacijama o narudžbama. Time zaključujemo da je ispoštovan princip S jer su klase odgovorne za samo jednu funkcionalnost.

- **Princip O** – Klase su dizajnirane tako da svako dodavanje novih metoda neće dovesti do toga da se cijela klasa mora mijenjati. Na primjer, nova vrsta plaćanja (npr. PayPal) može se dodati kreiranjem nove klase koja proširuje postojeći okvir za plaćanje, bez potrebe za izmjenom postojećih klasa. Isto tako, dodavanje novih tipova vozila ili vrsta korisničkih računa ne zahtijeva promjenu postojećih klasa, čime je omogućena proširivost sistema.

- **Princip L** – Naša aplikacija koristi apstraktne klase, kao što su Korisnik, Placanje, i Zaposlenik, što omogućava da se na mjestima gdje se koristi osnovni objekat može koristiti i izvedeni objekat. Na primjer, klasa Korisnik može biti zamijenjena sa klasama Administrator, Prodavac, Kupac, dok Placanje može biti zamijenjeno sa klasama poput Kartica ili Kredit. Ovo omogućava fleksibilnost u korištenju objekata bez narušavanja funkcionalnosti sistema.

- **Princip I** – Ovaj princip zahtijeva da svaki interfejs obavlja samo jednu vrstu akcije. U našoj aplikaciji imamo jasno definisane interfejse koji obavljaju specifične funkcije, kao što je interfejs IGenerisiIzvjestaj, koji koristi klasa Zaposlenik i obavlja samo jednu akciju to jeste generisanje izvještaja. Ovaj interfejs nije preopterećen nepotrebnim metodama, čime je ispoštovan princip segregacije interfejsa.

- **Princip D** – Ovaj princip zahtijeva da pri naslijeđivanju od strane više klasa bazna klasa bude uvijek apstraktna. U našoj aplikaciji, klase poput Osoba, Placanje i Zaposlenik predstavljaju apstraktne klase koje služe kao temelji za naslijeđene klase. Na primjer, Korisnik naslijeđuje klasu Osoba, a Kartica naslijeđuje klasu Placanje. Korištenje apstraktnih klasa omogućava fleksibilnost i smanjuje direktnu zavisnost između specifičnih klasa.