SOLID principi

1. **Princip pojedinačne odgovornosti (Single responsibility Principle)**

Svaka klasa treba imati samo jednu ulogu. Ovaj princip zahtijeva da svaka klasa ima samo jednu odgovornost, odnosno da klasa vrši samo jedan tip akcija kako ne bih ovisila o prevelikom broju konkretnih implementacija. Kod nas ovaj princip je za sad zadovoljen jer klase u našem sistemu sadrže samo getere i setere, konstruktore i destruktore. Gledajući dalje kroz program, naše klase trebale bi zadovoljiti ovaj princip.

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

Ovaj princip kaže da bi klase trebali biti otvorene za nadogradnju a zatvorene za modifikaciju. Ovaj princip zahtijeva da klasa koja koristi neku drugu klasu ne treba biti modificirana pri uvođenju novih funkcionalnosti, ili pri potrebi za mijenjanjem druge klase. U našem sistemu klasa koja bi možda imala problem sa ovim principom jeste klasa narudžba koja u jednoj narudžbi može primiti samo jedan automobil, te u slučaju da kupac hoće da kupi više od jednog automobila, ova klasa bi se trebala modificirati da može primiti više od jednog automobila. Također ovo može biti riješeno sa pravljenjem više narudžbi u sklopu jedne kupovine.

3. **Liskov princip zamjene (Liskov Substitution Principle)**

Svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima, bez da to utječe na ispravnost rada programa. Ovaj princip zahtijeva da nasljeđivanje bude ispravno implementirano, odnosno da je na svim mjestima na kojima se koristi osnovni objekat moguće skoristiti i izvedeni objekat a da takvo nešto ima smisla. U našem sistemu ovaj princip je zadovoljen jer jedine izvedene klase su klasa NoviAutomobil i PolovniAutomobil koje su izvedene iz klase Automobil. Ova klasa koristi u narudžbi i može se zamjeniti sa jednom ili drugom izvedenom klasom.

4. **Princip izoliranja interfejsa (Interface Segregation Principle)**

Bolje je imati više specifičnih interfejsa, nego jedan generalizovani. Ovaj princip zahtijeva da i svi interfejsi zadovoljavaju princip S, odnosno da svaki interfejs obavlja samo jednu vrstu akcija. U našem sistemu nemamo niti jedan interfejs pa 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. Ovaj princip zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktna. Kod nas jedino nasljeđivanje nalazimo kod klase Automobil, koja zadovoljava ovaj princip jer je ona apstraktna.