SOLID PRINCIPI

Single Responsibility Princip:

Ovaj princip kaže da svaka klasa treba imati samo jednu odgovornost. Napravili smo zasebne klase za svakog korisnika sistema, te klase koje opisuju opremu, trening i novost kao zasebne cjeline. Sve klase su sadrzane u jednoj kontejnerskoj klasi Teretana, koja ima ulogu da upravlja kompletnim sistemom. Vidimo da je princip zadovoljen.

Open/Closed Princip:

Ovaj princip kaže da klasa treba biti otvorena za nadogradnje, ali zatvorena za modifikacije. Iz apstraktne klase Korisnik mogu se izvoditi nove klase, bez da se išta mijenja u samoj klasi Korisnik, iz čega vidimo da je klasa Korisnik otvorena za nadogradnju. Kada su u pitanju klase koje kao atribute koriste instance drugih klasa, bilo kakva izmjena atributa ili metoda tih klasa neće se odraziti na klasu koja je koristi. Također, možemo dodavati nove atribute ili izvoditi nove klase iz već postojećih, bez potrebe za mijenjanjem trenutnih atributa i metoda postojećih klasa.

Liskov Substitution Princip:

Ovaj princip kaže da svaka osnovna klasa treba biti zamjenjiva svim svojim podtipovima bez da to utječe na ispravnost rada programa. U našem sistemu, klasu Korisnik nasljeđuju klase Clan, Trener, Recepcioner i Admin. Klasa korisnik se kao takva koristi u dvije metode klase Teretana, metodi posaljiPritupnePodatke i ukloniKorisnika. Ove metode vrijede za bilo koju od 4 klase izvedene iz klase Korisnik. Vidimo da je princip zadovoljen.

Interface Segregation Princip:

Za realizaciju našeg sistema nismo planirali korištenje interfejsa.

Dependency Inversion Princip:

Ovaj princip kaže da sistem klasa i njegovo funkcionisanje treba ovisiti o apstrakcijama, a ne o konkretnim implementacijama. U našem sistemu primjećujemo ovisnost klasa Clan, Trener, Recepcioner i Admin od klase Korisnik koja je apstraktna, pa je samim tim ovaj princip zadovoljen.