SOLID principi

1. Single Responsibility Principle – Princip pojedinačne odgovornosti

Ovaj princip nam govori da bi klasa trebala da ima samo jedan razlog za promjenu. Sve klase u modelu treba da sadrže gettere, settere i da po potrebi implementiraju interfejse ili metode naslijeđene iz apstraktnih klasa.

U našem sistemu smo uočili da klasa Tim nije zadovoljavala ovaj princip zbog metode *izračunajBodove()* koja ima zasebnu funkcionalnost. Stoga, da bismo "razdvojili poslove", napravili smo novu klasu *Bodovi* koja će imati 3 atributa tipa int(broj pobjeda, broj poraza, broj nerijerešenih) i metodu *izračunajBodove().* Klasa Tim i klasa Bodovi su povezane kompozicijom, a klasa Tim će umjesto navedena 3 atributa sadržavati jedan atribut tipa *Bodovi.*

Što se tiče ostalih klasa, svaka od njih ima po jednu funkcionalnost te sadrže samo getere i setere.

2. Open Close Principle – Otvoreno-zatvoreni princip

Ovaj princip kaže da bi klase trebali biti otvorene za nadogradnju a zatvorene za modifikaciju.

Naš sistem je dovoljno apstraktan da ne bi trebalo izvršiti nikakve promjene u slučaju dodavanja novih funkcionalnosti, stoga nije potrebno ništa mijenjati.

3. Liskov Substitution Principle – Liskov princip zamjene

Ovaj princip zahtijeva da podklase moraju biti zamjenjive baznoj klasi, tj. da bilo koja upotreba bazne klase omogućava upotrebu i izvedenih klasa, sa istim rezultatom. Naš sistem ne sadrži nasljeđivanja, stoga je ovaj princip zadovoljen.

4. Interface Segregation Principle – Princip izoliranja interfejsa

Ovo je single responsibility principle (S princip) za interfejse. Naš sistem ne sadrži interfejse, pa je ovaj princip zadovoljen.

5. Dependacy Inversion Principle – Princip inverzije ovisnosti

Moduli visokog nivoa ne bi trebali da zavise od modula niskog nivoa, oba bi trebalo da zavise od apstrakcija. Naš sistem ne sadrži nasljeđivanja, niti klase različitog nivoa, stoga je i ovaj princip zadovoljen.