SOLID PRINCIPI

SINGLE RESPONSIBILITY PRINCIPLE - PRINCIP POJEDINAČNE ODGOVORNOSTI

Princip pojedinačne odgovornosti zahtijeva da svaka klasa ima samo jednu odgovornost, odnosno da klasa vrši samo jedan tip akcija kako ne bi ovisila o velikom broju konkretnih implementacija.

Ukoliko pogledamo dijagram klasa vidimo da većina klasa implementira samo gettere i settere pripadajućih atributa ili radi sa listama koje posjeduje kao atribute što omogućava da ukoliko se promjene i dese one se dešavaju na jednom mjestu. Uzmimo primjer klasu User. Svi atributi ove klase su geteri i seteri te dodavanje nove mogućnosti neće zahtijevati izmjenu nigdje osim u toj klasi, što nam govori da je ovaj princip ispunjen.

OPEN CLOSED PRINCIPLE - OTVORENO ZATVOREN PRINCIP

Otvoreno zatvoren princip zahtijeva da klasa koja koristi neku drugu klasu ne treba biti modificirana pri uvođenju novih funkcionalnosti, ili pri potrebi za mijenjanjem druge klase

Sve klase imaju veliki broj metoda te dodavanje novih neće uticati na promjenu same klase i logike koja stoji iza nje odnosno dodavanje metoda neće zahtjevati izmjenu čitave klase. Većina klasa sadrži kolekcije drugih klasa tako da izmjena u nekoj klasi neće implicirati promjenu druge.

Primjer ovog principa možemo vidjeti kroz klasu Cart koja sadrži listu Product koja je u biti apstraktna klasa iz koje su izvedene klase Accessorie, Clothing i Shoes. Ukoliko bismo odlučili dodati novu klasu koja ce biti izvedena iz Product npr. SportsWear nikakve promjene ne bi bile potrebne u klasi Cart. Nasuprot tome da Product nije apstraktna klasa te da u Cart imamo kolekcije Shoes, Accessories i Clothing, pa ukoliko kreiramo novu klasu npr. SportsWear neophodna bi bila i izmjena u Cart što bi narušilo open-close princip.

LISKOV SUBSTITUTION PRINCIPLE - LISKOV PRINCIP ZAMJENE

Liskov princip zamjene zahtijeva da nasljeđivanje bude ispravno implementirano, odnosno da je na svim mjestima na kojima se koristi osnovni objekat moguće iskoristiti i izvedeni objekat a da takvo nešto ima smisla.

Dijagram posjeduje samo jednu apstraktnu klasu, a to je Product iz koje su izvedene klase Shoes, Accessories i Clothing. Jasno je da se na svakom mjestu, gdje je potrebno koristiti neku od ove tri klase, koristi apstraktna klasa iz koje su iste izvedene i da je upotreba bilo koje od izvedenih klasa na mjestu apstraktne moguće.

Primjer upotrebe apstraktne klase jeste recimo u klasi Cart gdje se koristi lista Products. U toj listi je moguće čuvati bilo koju od izvedenih klasa zbog čega je ovaj princip i ispunjen.

INTERFACE SEGREGATION PRINCIPLE - PRINCIP IZOLIRANJA INTERFEJSA

Princip izoliranja interfejsa zahtijeva da i svi interfejsi zadovoljavaju princip pojedinačne odgovornosti, odnosno da svaki interfejs obavlja samo jednu vrstu akcija.

U našem sistemu ne postoje interfejsi, o ovome principu ne možemo ni diskutovati te smatramo da je ispoštovan.

DEPENDENCY INVERSION PRINCIPLE - PRINCIP INVERZIJE OVISNOSTI

Princip inverzije ovisnosti zahtijeva da pri nasljeđivanju od strane više klasa bazna klasa uvijek bude apstraktna. Razlog za ovo je što je teško koordinisati veliki broj naslijeđenih klasa i

konkretnu baznu klasu ukoliko ista nije apstraktna, a da pritom kod bude čitak i jednostavan za razumijevanje.

Jedina klasa kod koje se javlja nasljeđivanje je klasa Product. Ona je navedena kao abstract odnosno nemoguće ju je instancirati. Iz nje su izvedene klase Shoes, Accessories i Clothing koje je moguće instacirati, stoga se smatra da je ovaj princip ispoštovan.