SOLID principi

* S – princip pojedinačne odgovornosti

Ovaj princip je ispoštovan. Iz metoda klasa (koje su u drektnoj vezi sa bazom podataka ili sa XMLom) vidimo da te klase imaju samo jedan razlog za izmjenu klase, a to je izmjena atributa. Izmjena u bazama ili u formata za izvještaj neće utjecati na smu klasu. Metode kojima će se obavljati rad s bazama ili XMLom nalazit će se u interfejsima koji nisu dodani na dijagram zbog svoje specifične primjene, kao i samog referenciranja '1 na 1' sa određenom instancom klase. Uz sve to, uzrok je svakako i estetska konzistentnost.

* O – princip otvoreno zatvoreno

Ovaj princip je ispoštovan uzimajući u obzir prethodnu konstataciju da će metode za rad sa vanjskim aplikacijama biti smještene u odvojene interfejse. S druge strane, posmatramo li interfejs klase HotelskePogodnosti, vidimo da dodavanjem neke posebne klase gostiju, morali bismo (u skladu s tim kakva bi to klasa gostiju bila) dodati neke nove metode u ovu klasu, što obezbijeđuje da je sistem otvoren za nadogranju.

* L – Liskov princip zamjene

Ovaj princip je također ispoštovan jer su sve generalizacije korektno odrađene. Uzmemo li primjer klase Osoba i klase izvedene iz nje (Administrator, HotelskoOsoblje, Gost), vidimo da instanca svake od tih klasa mora imati atribude ime, prezime i brojLičneKarte. Kada bismo dodali klasu PravnoLice izvedenu iz klase Gost, tada bi ovaj princip bio narušen, jer bi se u tom slučaju zahtijevalo od instance klase PravnoLice (što bi se odnosilo na predstavnike nekih firmi, od čega bi bilo važno ime firme) zahtijevalo da ima i pobrojane atribute, što za ovu klasu ne bi imalo smisla.

* I – Princip izdvajanja interfejsa

Gledajući klase sa dijagrama, vidimo da sadrže samo propertyje, odnosno metode get i set kojima se postavlja ili očitava vrijednost atributa. Dodavanjem interfejsa na zadane klase, nastoji se odvojiti metode koje su pod utjecajem korisnika i metode koje su posrednik u operacijama (što najčešće podrazumijeva skraćivanje koda). Planirano dodavanje interfejsa, podrazumijeva da se izdvoje radnje koje su pod direktnim utjecajem korisnika i tako se ispoštuje i ovaj princip.

* D – princip inverzne ovisnosti

Ovaj princip nije ispoštovan. Najočiti primjer toga je taj što se vrši generalizacija iz neapstraktne klase Soba. Grupacija je konkretna klasa (iako je statička s obzirom da je aplikacija na nivou jedne instance ove klase) i mora biti agregacijom vezena sa hotelima. Hoteli su također konkretna klasa (jer nismo vršili nasljeđivanje iz te klase) i kao takva mora biti u agregaciji sa konkretnom klasom. No čak i kada bismo klasu Soba označili kao apstraktnu, dvije konkretne klase Grupacija i Hotel bi bile povezane asocijacijom (ili u vom slučaju agregacijom), što narušava ovaj princip.

P.S. Ekipa PrezimeNijeBitno se izvinjava na narušavanju ovog principa i moli da nam to ne uzmete za zlo. Drugi put ćemo biti bolji. Hoćemo Boga mi!